A Floor Control Server in a Distributed Conference Service

Aila H.Koponen

Helsinki, 18 April 2008
M.Sc. Thesis
University of Helsinki
Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion - Faculty	Laitos – Institution - Department
Faculty of Science	Department of Computer Science

Tekijä – Författare - Author
Koponen, Aila Helena

Työn nimi – Arbetets titel - Title
A Floor Control Server in a Distributed Conference Service

Oppiaine – Läroämne - Subject
Computer Science

Työn laji – Arbetets art - Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Master's Thesis	April 2008	76 + 20 pages

Tiivistelmä – Referat

The conferencing systems in IP Multimedia (IM) networks are going through restructuring, accomplished in the near future. One of the changes introduced is the concept of floors and floor control in its current form with matching entity roles.

The Binary Floor Control Protocol (BFCP) is a novelty to be exploited in distributed tightly coupled conferencing services. The protocol defines the floor control server (FCS), which implements floor control giving access to shared resources. As the newest tendency is to distribute the conferencing services, the locations of different functionality units play an important role in developing the standards. The floor control server location is not yet single-mindedly fixed in different standardization bodies, and the debate goes on where to place it within the media server, providing the conferencing service. The thesis main objective is to evaluate two distinctive alternatives in respect the Mp interface protocol between the respective nodes, as the interface in relation to floor control is under standardization work at the moment.

The thesis gives a straightforward preamble in IMS network, nodes of interest including floor control server and conferencing. Knowledge on several protocols – BFCP, SDP, SIP and H.248 provides an important background for understanding the functionality changes introduced in the Mp interface and therefore introductions on those protocols and how they are connected to the full picture is given. The actual analysis on the impact of the floor control server into the Mp reference point is concluded in relation to the locations, giving basic flows, requirements analysis including a limited implementation proposal on supporting protocol parameters.

The overall conclusion of the thesis is that even if both choices are seemingly useful, not one of the locations is clearly the most suitable in the light of this work. The thesis suggests a solution having both possibilities available to be chosen from in separate circumstances, realized with consistent standardization. It is evident, that if the preliminary assumption for the analysis is kept regarding to only one right place for the floor control server, more work is to be done in connected areas to discover the one most appropriate location.

CR -classification: C.2.1, C.2.2

Avainsanat – Nyckelord - Keywords
BFCP, H.248, MRF, floor control, conferencing, IMS, Mp interface

Säilytyspaikka – Förvaringställe – Where deposited
Kumpula Science Library, serial number C-

Muita tietoja – Övriga uppgifter – Additional information

To my adorable daughters Siiri and Iida, and to Jouni, my most valuable support.

Acknowledgements

This Master's thesis was written at LM Ericsson, Jorvas, Finland.

Primarily I wish to thank my instructor Gonzalo Camarillo for providing me with an interesting subject and for the most valuable guidance during writing and Teemu Arvonen for great support, endurance and advices.

I would like to thank my supervisor at the Department of Computer Science at the University of Helsinki, Senior Researcher Markku Kojo, for good instructions and flexibility, and also professor Jussi Kangasharju for supporting in correcting and final evaluations of this thesis.

My gratitude goes to Fredrik Åberg for constructive ideas and patience, Oscar Novo and Edgar Ramos for listening to my questions and giving the most needed help and to all my colleagues at Ericsson for their help and suggestions.

I want also to thank NomadicLab as well as Mobile Media Gateway Integration & Verification Department for support and for giving me a chance, time and space to write this thesis.

Finally, my warmest thanks go to my family for helping me through tough times and to all my friends who kept me going with their positive attitude.

Kirkkonummi, April 18, 2008

Aila H. Koponen

Contents

Index of figures

Acronyms

ABNF	Augmented Backus-Naur Form
ALF	Application Level Framing
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
AS	(SIP) Application Server
3G	3rd Generation mobile telephony
3GPP	3rd Generation Partnership Project
BFCP	Binary Floor Control Protocol
BNF	Backus-Naur Form
CS	Circuit Switched
CSCF	Call Server Control Function
FCFS	First-Come-First-Served
FCS	Floor Control Server
FIFO	First-In-First-Out
H.248	ITU-T Recommendation H.248
H.323	Umbrella including also H.225.0, H.245, the H.450-series and H.460-series
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IM(S)	IP Multimedia (Subsystem)
IP	Internet Protocol
IPBCP	IP Bearer Control Protocol
ISC	IMS Service Control, API used to connect an application server to IMS
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector of ITU
MCU	Multipoint Control Unit
MG, MGw	Media Gateway
MGC(F)	Media Gateway Controller (Function)
MEGACO	Media Gateway Control protocol by IETF
MIME	Multipurpose Internet Mail Extensions
MRB	Multimedia Resource Broker
MRF	Multimedia Resource Function
MRFC	Multimedia Resource Function Controller
MRFP	Multimedia Resource Function Processor
MS	Media Server
MSC-S	Mobile-services Switching Centre Server
MTSI	Multimedia Telephony Service for IP Multimedia Systems
PLMN	Public Land Mobile Network
QoS	Quality of Service
RFC	Request For Comments
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTP/AVP	Audio and Video Profile for RTP
SAP	Session Announcement Protocol
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol (older: Session Invitation Protocol)

TDM	Time Division Multiplex
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Type-Length-Value
UA	User Agent, Client (UAC) or Server (UAS)
UDP	User Datagram Protocol
UE	User Equipment
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoIP	Voice over IP
WWW	World Wide Web
XML	eXtensible Markup Language

1 Introduction

The task of a conferencing system in a network is to provide means to arrange a meeting for a group of people, wishing to communicate with each other, in spite of geographical distances. The definition of conferencing traditionally stands for conferences, realized with multicast media groups or models using distributed control with direct signaling relationship between all conference participants. The recent approach is to use centralized conferencing control where all the participants connect straight to the conferencing system. The new systems make decisions according to user requests on participation and conference floor control, informally expressed as control over "right to speak", in predefined manner or on demand by consulting conference participants when needed. The target under discussion in this thesis is the generally accessible IP Multimedia network [3GPP07b, 3GPP07d] conferencing system [3GPP07c] aimed to be deployed in the future in a standardized way to access the conference resources with assistance of floor control.

The conference floor is a concept, denoting a temporary permission given to a conference participant to access or manipulate a specific shared object, resource or set of resources in the context of a conference. Floor control realizes advanced conference control by enabling applications or conference participants to gain safe, mutually exclusive or non-exclusive input access to floors [COD06]. Typically, a floor participant requests floor and information on floors from a *floor control server* [COD06], located in the media server which provides connectivity, call control and multimedia services for users. Floor control introduces also a logical entity, managing one floor at a time – the floor *chair*. The actions performed by a chair remind of the duties of a corresponding person in the real world - a chair can grant, deny or revoke a conference participant's request to gain the floor. A chair for a given floor control transaction may also assume a different role, e.g. floor participant, for a different transaction. The roles of the floor chair and the floor participant in a conferencing system are defined on a transaction-by-transaction basis. Floor control is an optional element in conferencing applications.

A floor control server implements tightly coupled conferencing control - towards the conference participant it uses the floor control protocol [COD06] recently defined by IETF. The floor control protocol does not implement traditional conference control, meaning session set-up, tear-down and resource association to a specific conference floor. Signaling protocol and session description protocol are required to deliver conference connection information for both endpoints of the conference relationship. The connection information includes, e.g., the data according to which the conference user can acquire and release the direct connection to floor control server. This thesis

introduces the protocols and their core features needed for formation of the connection to floor control server.

Several standardization bodies specify a supplementary conference service framework. Multimedia Telephony Service for IP Multimedia Systems (MTSI) specifies it under the name CONF, and Internet Engineering Task Force (IETF) introduces a centralized conferencing system called XCON. The 3rd Generation Partnership Project (3GPP) on its behalf describes matching conferencing architectures. These architecture and service descriptions are employed in current conferencing service implementations. Conference floor control is, in the context of involved organizations and evolvement of these services, a well defined feature, while the location for the implementation of the corresponding server is still under discussion. A floor control server in the IP Multimedia Network can be located in different places within a distributed media server, and all the aspects on the environment are not yet clearly specified. The internal functional split of floor control between media server nodes is still an implementation decision and is currently under discussion within 3GPP. Some trials to realize floor-controlled implementations on the current IP Multimedia Systems (IMS) have been made in spite of the condition of the specifications. The location for the floor control server and other affiliated functions must then be chosen, an example of such work is presented in the article "A Distributed IMS Enabled Conferencing Architecture on Top of a Standard Centralized Conferencing Framework" [Buo07], introducing an advanced distributed conferencing framework implementation.

The debate whether to put the floor control server into the control plane or to the user plane side of the media server is still ongoing. The difficulty in this is that there are multiple reasons why the location of the floor control server should be in one place or another. The 3GPP considers performance, load balance within the media server, extensibility of the floor control protocol, a number of open issues brought by distributed conferences, etc., affecting the floor control server location. Still none of these issues is a single decisive factor regarding the topic. The open questions around floor control is starting to take form, but no study has yet been made on how exactly the control plane to user plane network interface changes according to the server placement. Depending on the location of the floor control server, the interface has distinctively different impacts. To address this problem is the main challenge of this thesis.

One of the significant unfinished standardizing issues is defining floor control support on the control plane to user plane interface within the media server. The interface is denoted as the Mp reference point or Mp interface. The study of the behavior brought by the functionality addition from floor control to the Mp interface is the most important objective of this thesis. In addition to the main objective, the problem

setting of this thesis concentrates on effects brought by some related issues; distribution of conference media handling to other media servers within the IP Multimedia network and manipulation of the media streams for a participant (e.g. "shutting down" a participant) ruled by floor control. We studied also the possibility to determine the floor control server location on-the-fly or semi-permanently within the media server according to decisions based on network configuration or locally during conference set-up. One of the goals of the thesis is to contribute an implementation proposal on floor control support in the Mp interface.

The results for the study are achieved by exploring involved protocols and the Mp interface between the distributed media server network components, considering them as possible locations for the floor control server. The requirements for floor control regarding the protocols involved and consequences of choices are considered in respect to each site selected. Participant media manipulation impacts and the support for further distributing the handling of different multimedia types are separately studied in relation to each of the thesis objectives. The evaluation is done in respect to protocol primitives and parameters. The analysis prepared on floor control requirements gives a suggestion on functionality on the Mp interface. The implementation proposal allows different options for floor control locations regarding a single conference or network setting. The combinations for locations give a single floor control server in the control or user plane of the media server or multiple floor control servers in the user plane in different media servers. Those are analyzed with viewpoint set by cascaded conferencing models [Nov06]. The proposal supports several functionality features - floor control message tunneling in Mp interface, distributed media mixing, choosing floor control algorithm, conference moderator and chair, and floor holder limiting. The proposal gives the possibility to perform participant media manipulation in any level chosen by the control plane.

The scope of the thesis mainly focuses on tightly coupled conference control in IP Multimedia Systems environment. The medias the conferences may use are audio, video and session based messaging. Affiliated protocol interworking is shown when a user is connecting to a floor control server, when feasible. The protocols studied do not cover conference policy control even if it is needed for the actual service. The policy in context to conference and floor control means the authorization and limitation of access to conference resources and floors depending on predefined rules by the network operator or conference arranger. The policy control and implementation of floor control server will not be included in the scope of the thesis.

The rest of the thesis is structured as follows. Section 2 contains introduction to the conferencing system at hand to give a basic comprehension of the problem scenery. Section 3, as a portrait of the network environment, examines the nodes of interest, the

floor control server, protocols involved in the conferencing system, and depicts establishment of the floor control server connection towards a conference participant. Section 4 gives an overview on the novel protocol for floor control. The gateway control protocol used in Mp interface separating the control plane and user plane in the media server is introduced in section 5. The Mp interface is explored in the section 6 according to the thesis objectives, giving also an implementation proposal for floor control support in the interface, and finally the implications of the findings obtained in the study with standardization aspects are discussed in the section 7.

2 Conferencing system

The assignment set for a conferencing system is to offer a service implementing a meeting floor for group communication. Without knowing the limitations of modern distributed conferencing services, an average user would expect that utilizing floor control would be a standard procedure. This might be true for certain proprietary conferencing implementations, which tie the participants to specific solutions or network configurations. One of the non-standard solutions for the area is presented, e.g., is the paper "Isabel: An Application for Real Time Collaboration with a Flexible Floor Control" [Que05]. To complete generally available and easily deployable services for floor-controlled conferencing, we need a standardized way to access the conferencing resources. In order to accomplish this, we must first set the basic characteristics of the conferencing system to support our aims. In this section we first describe the different general conferencing models, architecture chosen for the model interesting for the scope of the thesis, and then give a short introduction to the common operations.

2.1 Conferencing model

The conferencing systems can be roughly divided into three groups: *Loosely coupled*, *fully distributed* and *tightly coupled* conferencing systems [Ros06].

In a *loosely coupled conferencing model* there is no conference server or a central point of control. The media is distributed using multicast media groups and the signaling relationship is missing between the participants[1]. The conference participants learn gradually about the other participants in the course of received media packets.

A *fully distributed multiparty conferencing model*, sometimes called also the *full mesh model*, does not have a central point of control either. All the participants keep signaling relationships with each other, e.g. using SIP. Since the conference control is fully distributed for the participants, each and every one sends a copy of their media packet to all other participants through unicast. An example of the model and a protocol designed for it is introduced in the conference paper "A Protocol for Reliable Decentralized Conferencing" [LeS03].

The *tightly coupled conferencing model* [BBL07] uses central point of control, the focus. It keeps a direct peer association as a call signaling relationship with each participant,

[1] Participant is an entity that acts as a conference floor participant, as a media participant, or as both.

which always results into a star topology. In this context a call is a channel or a session used for media streams. This model is the one to be discussed in this thesis.

2.2 Tightly coupled conferencing architecture

In this subsection we give a brief introduction to the elements and the conference information model. The tightly coupled conferencing architecture consists of following elements: focus, *media mixer*, *conference notification service*, *participant*, *conference policy server*, and *conference policy* [Ros06]. In addition, IETF defines an element, namely a *conference factory* [JoL06, BBL07] – a logical entity that generates unique *Uniform Resource Identifiers* (URI) [Ber05] to represent and identify the conference *focus* which controls the conference.

The call signaling topology is depicted in Figure 2-1. The media graph a conference, i.e. what kind of medias are deployed and how the media sessions are connected in the sessions forming the multiparty call, is centralized. This means, that even if the media sessions are established between the mixer and each one of the participants, the call signaling relationships (in our case Session Initiation Protocol (SIP) [Ros02] dialogs[2]) are formed between the participants and the conference focus separately. Combining the separate media streams of same type from different participants into a single stream is done by the media mixer. It distributes the resulting stream then to its destination(s).

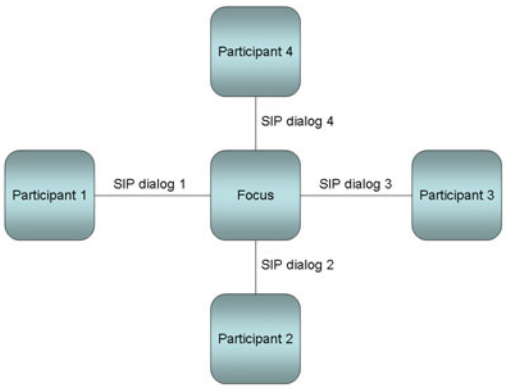

Figure 2-1 Tightly coupled conference [Ros06]

A *conferencing client* is implemented in the user equipment at the conference participant and provides him access to the conferencing system. The conferencing system, shown in the Figure 2-2, communicates with the conferencing client with various protocols. The Floor Control Server (FCS) and the Binary Floor Control Protocol (BFCP) [COD06] offer the floor control for the *floor control client*. The conferencing client

[2] A SIP dialog is described as a relationship between two peers that persists for some time and is uniquely identified.

consists of four dedicated protocol clients, access protocols (except BFCP) and requirements of which are out of scope of this thesis and are not discussed here further.

A conference notification service is always implemented in the conferencing system and it can be provided by the conference focus. The notification service takes subscriptions from users and delivers notifications to authorized parties accordingly. The notifications include conference instance state changes e.g. on participants involved in joining and leaving the conference. Furthermore, identity suppression is made available for participants wishing to stay anonymous.

The functional elements accessing the centralized conference information, in form of a *conference object*, are *foci*, *conference control server*, *notification service* and *floor control server*, as illustrated in the Figure 2-2. The notification service is different from the servers in that its functionality is more subject to free implementation and it is not necessarily to be realized within the media server domain.

The conference policy server is the interface between the conference client and the conference policy which consists of rules and guidelines for the conference operation, e.g. user access list for a conference or time-of-day based rule sets. The typical access to a conference policy server are voice- or web applications, not included in the thesis scope.

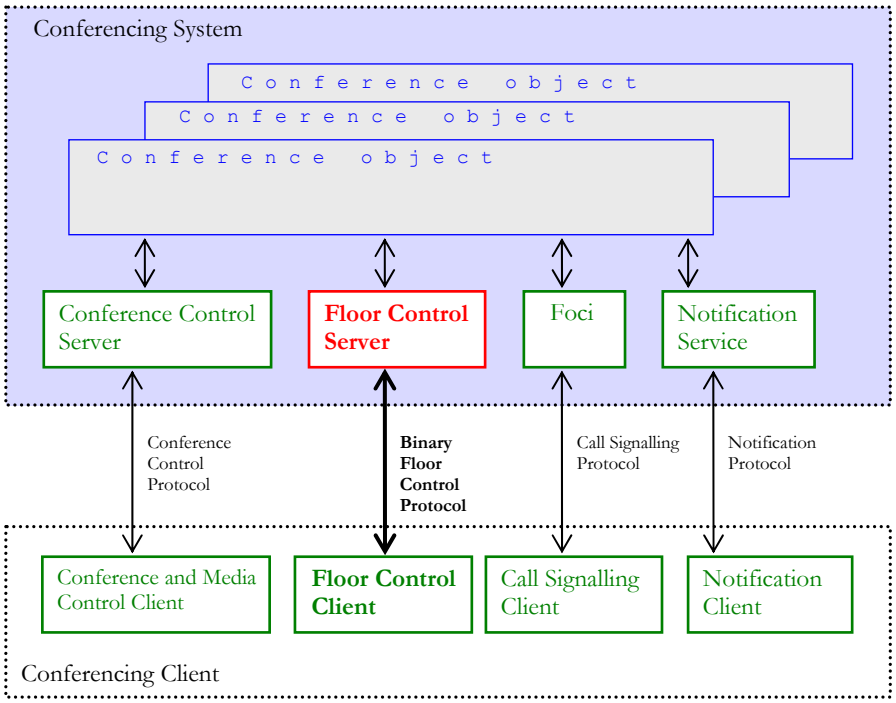

Figure 2-2 Conferencing System logical decomposition [BBL07]

A conference object has direct one-to-one relationship to a conference focus. The conference object holds the mapping to interfacing functional elements with other conference information. The core information definitions in the conference object that are used in any conference include conference capabilities, membership, roles, call signaling and media status relevant to different stages of the conference. The conference object itself is of specific extensible *conference information type*[3], shown in Figure 2-3. Using the specific mixing details, floor controls and other data provided by the model, the enhanced conferencing features can be supported, e.g. changing the participant's states by manipulating the mixer and the media streams via focus.

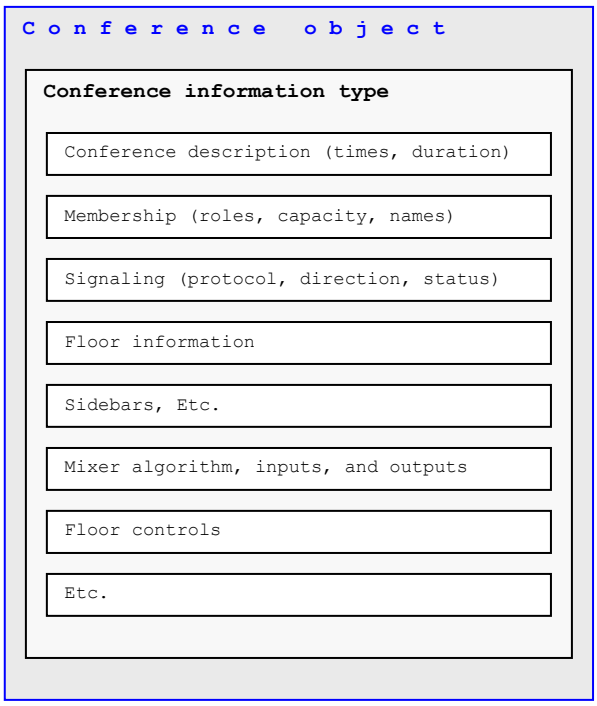

Figure 2-3 Conference object type decomposition [BBL07].

2.3 Common conferencing operations

Traditionally a conference user does have some say in the participation to a conference. He has the possibility to join or leave the conference and to use a conference notification service to get information on who is around in form of indications depending on the notification service implementation. The participant, possibly

[3] The Extensible Markup Language (XML) schema on the type and details of the usage can be found in the centralized conference information data model [Nov07].

assuming a separately agreed role of a moderator or conference chair, may even be able to manipulate the users' media streams, e.g., by including and excluding, (i.e. "shutting up") a participant.

The common conferencing operations for the tightly coupled, centrally controlled and floor-controlled conferences may include all of those features. The connection to the Conferencing System is done on initiative of a user to join or a request to add someone else to the conference. These are called *first-party addition* and *third-party addition* according to who initiated it. In *first-party addition* the participants are contacting the conference system by themselves and in high level the signaling for the set-up is done similarly as for the first participant. If the conference participants wish to add more other users, the procedure is called *third party addition*, and there are two ways to do it. A participant can be requested to initiate a dialog to the to the conferencing system in order to add him as the new participant. The other possibility for a participant to add another user to the conference is to send a request (informing of the new participant) straight to the conference focus. The focus will then initiate a dialog towards the new user.

Once the connection using a defined media graph is made, the notification service can be deployed. If the floor control is well implemented for the user equipment, the need for the notification service can be somewhat reduced since some floor control procedures partly may cover the offered notification services. When wishing to end the conference connection, the participant can exit himself or the participant can be kicked out from the conference by another participant, given the right permissions in the conference policy. These ways to leave the conference are called *first party tear-up* and *third party removal*, analogous to the user additions.

Floor control does not change the possibilities to use the common operations. Joining and leaving the conference are done in the same way but the other operations can be performed during a conference as well. The common conferencing operations are realized with Session Initiation Protocol (SIP) signaling in the conferencing model we are focused on, while the floor control procedures are a part of advanced conferencing operations accomplished with BFCP.

3 Tightly coupled conferencing in IP Multimedia network

In this section we describe the environment in relation to the topic of the thesis. We overview the service and network structure with nodes of interest, specially the floor control server (FCS). The procedure and protocols needed for user connection establishment towards the floor control server are also introduced.

The IP Multimedia Subsystem (IMS) in 3rd Generation (3G) mobile telephony architecture can be characterized with the next sentences: "Aimed at person to person communication, IMS is the only standardized way to deliver IP-based services that are enabled by one common core and control for all types of networks. It provides users with attractive communication services over multi-devices across multi-access technologies" [Eri07a].

In order to provide Internet services for cellular users the IMS network is aimed to combine the better parts of the two worlds [CaG05]. The characteristic factors, openness from Internet side and ubiquity brought by mobility that the cellular networks represent, are a combination predicted to be the key to the widespread success. Until these days many of the services provided for cellular users have been just pale versions of the ones offered for Internet users. Services have emerged in horizon for the mobile consumer including World Wide Web (WWW), email, instant messaging, presence, Voice over IP (VoIP), video conferencing and shared whiteboards to mention some of the most appreciated ones. Now among the latest prospects is advanced conferencing service with centralized handling of user control over "right to speak".

Until now, conferencing in IMS has reminded of the services provided by the fixed legacy telephony. Once the individual conference has been set up, no control over who has the right to use the conference floor has been available in a consistent fashion. The protocols designed for this purpose have been more or less proprietary or inadequate. The current definition of floor control [COD06] comprises the user initiated efforts to manage the "right to speak" during a live conference in a controlled way to gain and release the right for using the floor, consisted of a media or different medias, e.g. video and audio. The floor control functionality includes the possibility to provide indications for the participant user equipment (UE) on conference statuses related to the functionality and to use a conference chair or a moderator, logical roles that remind the ones exercised in the real world. Since the standardization efforts are now jointly focused towards the tightly coupled user controlled conferences, the 3G through 3G Partnership Project (3GPP) by means of IMS tries to glue its requirements together to

provide an environment that could be used for service creation of multimedia conferencing systems using floor control defined by the Internet Society.

3.1 Network structure

We introduce now the architectural components in the IMS network from floor control point of view. The full IMS network picture is depicted in the Figure 3-1.

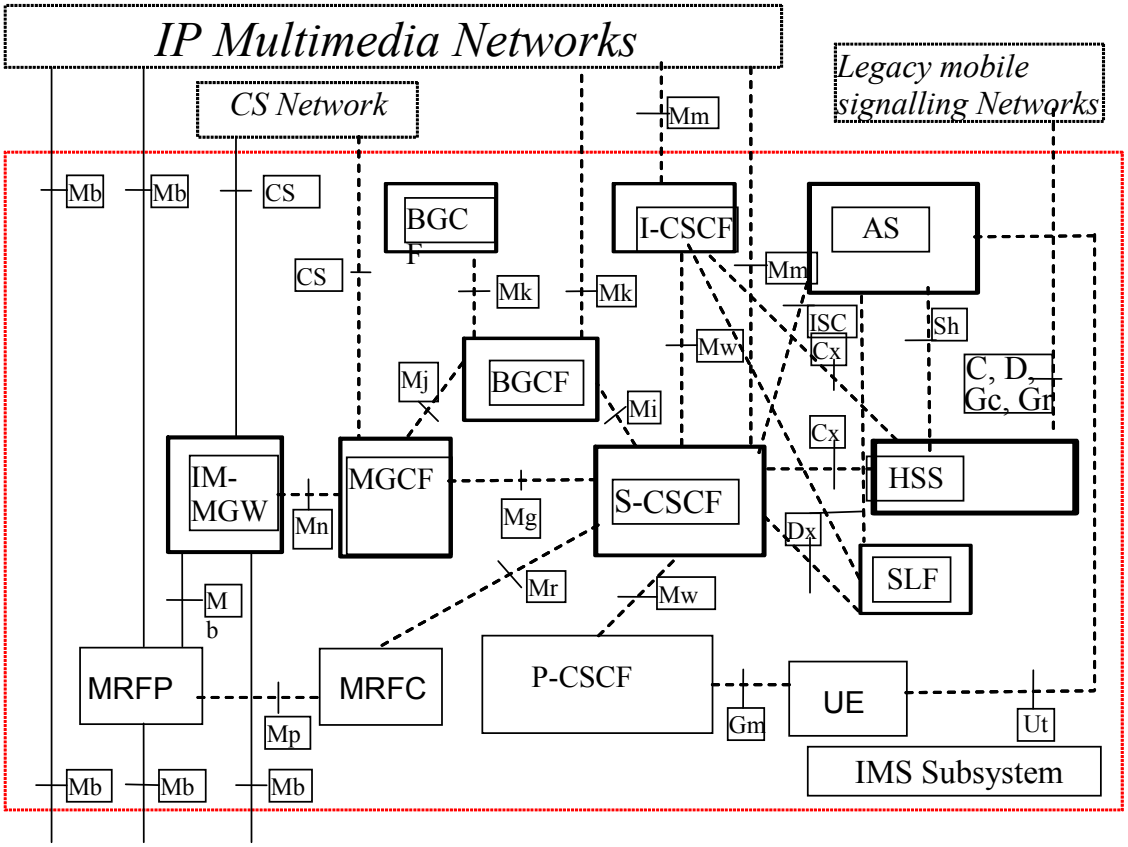

Figure 3-1 Reference Architecture of the IP Multimedia Core Network Subsystem [3GPP07b]

Most of the nodes in the figure are not relevant for examining the study subject but are shown merely to give understanding on the positions of the nodes in the network. The notable nodes are the Multimedia Resource Function Controller (MRFC), the Multimedia Resource Function Processor (MRFP), and the application server (AS) described here briefly from the thesis perspective. The description of IP Multimedia Media Gateway (IM-MGW) with Media Gateway Control Function (MGCF) is given also in short. For more information on the other nodes and architecture, see Public Land Mobile Network (PLMN) architecture and configuration [3GPP07d] and IMS network descriptions [3GPP07b] [Eri07a] [Nov06].

In Figure 3-2 the green highlighted area comprises the Multimedia Resource Function (MRF) as the distributed media server domain, media server here denoting a complex of MRFC in signaling plane and MRFP in media plane.

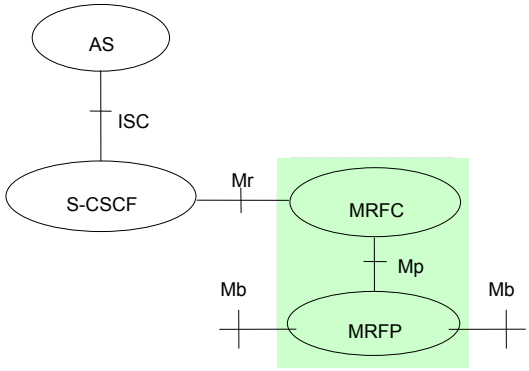

Figure 3-2 Architecture of Multimedia Resource Function as media server [3GPP07f]

From floor control point of view the Mb interfaces represent the payload and user data flow directions towards conference participants and IP Multimedia network, in our case including the floor control messages by means of Binary Floor Control Protocol (BFCP). The media server's Mr interface represents the control flow direction towards the application server (AS)[4]. Figure 3-2 shows the Serving Call Session Control Function (S-CSCF) between the application server and the MRFC, but the 3GPP approach today is to consider the conferencing application server (AS) interfacing the conference participants for call control signaling and MRFC as a single AS/MFRC[5] functionality. It will not affect exploring the Mp interface within the media server.

The media server is shown as functions in Figure 3-3, reflecting the alleged FCS location by 3GPP. The functional split between media server nodes is still not clear or self-evident, even if according to the current 3GPP specifications FCS should be located in the MRFP. Figure 3-3 shows in the AS/MRFC the conference and media conference policy servers and the conference notification server, hidden behind the conference focus. The floor control server interacts with them via the conference focus, if needed.

---

4 Application server, one of *SIP AS*, *OSA AS* or CAMEL IM-SSF, offers IM services and resides either in the user's home network or in a third party location, meaning a network or simply a stand-alone AS. An Application Server may influence and impact the SIP session on behalf of the services supported by the operator's network. An AS may host and execute services [3GPP07d].

5 The picture shows more nodes in between but logically the AS/MRFC comprises the controller part of the Multimedia Resource Function (MRF) and the SIP conference application server (AS) where the existence of serving call server control function (S-CSCF) is not relevant for floor control functionality.

In this thesis, the assumption on the floor control server location follows the assumption Figure 3-3. In the subsection 6.3 ”Floor control server in Multimedia Resource Function Controller” the assumption is contradicted. As comparison, the “Requirements for Floor Control Protocols” Request For Comments (RFC) 4376 [Kos06] states that the FCS is typically located with the conference focus.

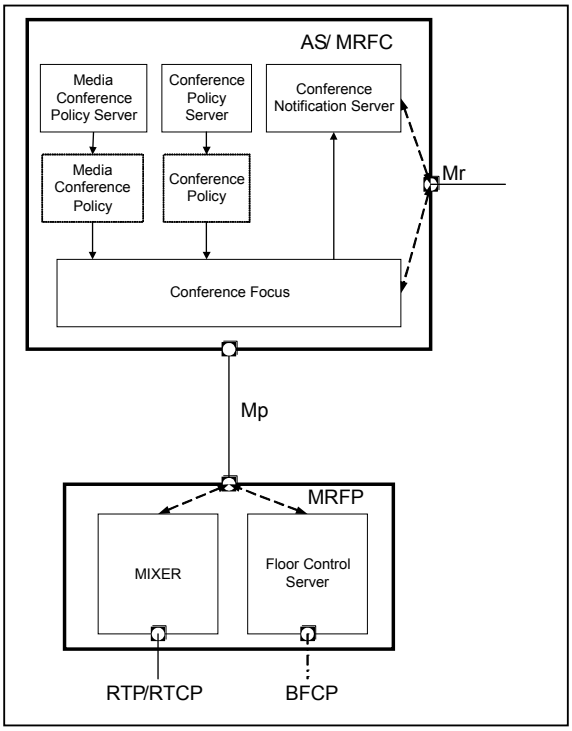

Figure 3-3 Conference logical functions spread in media server [3GPP07e]

The functionality of the MRFP includes the control of the bearers attached to it containing establishing, releasing and modifying connections, congestion and quality of service control, and security measures. The conference floor control is handled via Mb interfaces. Furthermore, the MRFP possesses all the media stream resources to support the user-plane services requiring media processing. The typical MRFP responsibility area comprises also: mixing of incoming media streams, providing announcements, media analysis, and actual stream processing. Stream processing in MRFP includes, e.g., audio transcoding between 3G and fixed telephony network subscribers.

The MRFC tasks related to floor control includes typical control plane services: interpreting and acting upon the control information from Mr interface or application server and controlling the MRFP media resources via the Mp interface, identified by the Mp *reference point*[6].

[6] *Reference point* is a conceptual point at the conjunction of two non-overlapping functional groups.

The Mp interface has open protocol architecture[7] and it fully complies with the gateway control protocol H.248 [Itu05] standard. The actual protocol to be deployed in the reference point is not specified in the current release of the 3GPP specifications [3GPP07b, subsection 4.7].

The IM-MGW handles in the IP Multimedia network the characteristic media gateway functions, reminding the MRFP functions. A media gateway converts media from one type of network to a format of another type of network. It can, e.g., process audio and video, perform media translations, play audio or video messages, perform other interactive voice functions and it can be used for conferencing services. The Media Gateway Control Function (MGCF) performs the media gateway controller duties. Those include handling the call state related tasks and controlling the connections for media gateway and its channels. The interface between the media gateway and media gateway controller, the Mn reference point, deploys the H.248 protocol and is highly similar to the Mp reference point. The Mn reference point is better standardized but does not have requirements for floor control support. That is though subject to change whenever and if the floor control is introduced to media gateways due to MGw and MRFP synergies.

3.2 Floor control server

The floor control server (FCS) is a part of the Conferencing System and it handles floor control during a live conference by communicating with conference focus and using Binary Floor Control Protocol (BFCP) towards the conference participant via Mb interface. In Figure 3-4 the logical entities, the floor chair, floor participants and the server, are depicted along with the overall functionality involved. The interactions of clients towards the floor control server include also requesting information about floor requests, floors, participants and capabilities of the server and receiving responses to these requests. In addition the server supports a *ping* -like transaction called "Hello" for clients to check the liveliness of the server.

The floor control server is a logical entity maintaining the floor states, including existence, chairs and holders of the floors. During a tightly-coupled conference the access to resources by floor manipulation is done by the floor control server. BFCP protocol provides a means for floor requests and responses, server notifications and decision messages between conference entities on an existing conference. FCS takes part in all of these operations. The conference creation and termination are done by other means outside the scope of the FCS. The server merely is kept up-to-date of those changes to the centralized conferencing data.

[7] This means that protocol extensions (as packages) may be defined for the interface.

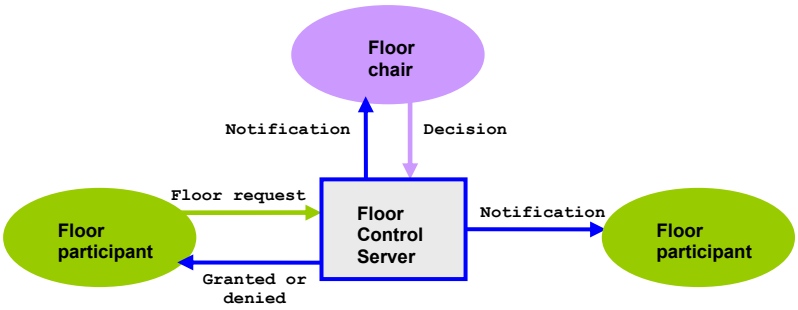

Figure 3-4 Functionality provided by BFCP [COD06]

The first encounter of the conference participant with Conference System is an invitation to or from a SIP application server (SIP AS) in an AS/MRFC requesting for floor-controlled connection. The BFCP connection parameters are propagated between the participant and the application server in session descriptions. The user parameters are transmitted via the Mp interface in addition to the normal control data flow coming from setting up and releasing the user connections.

Floor Participant is a conference participant entitled to request "right to speak" in form of a floor. Floor participant can request the floor from the server and will receive a *grant* or *denied* message back. Also cancelling a request or releasing the floor is possible. Here the floor release means that the floor can be requested again within the conference once it has been released. The participant exits the conference by means outside the FCS scope, in our case using SIP, but the information on it is updated to the conference object[8] so the server may obtain the information. The floor control server of a conference may also perform other logical roles (e.g., floor participant) in another conference.

Floor Chair is a conference participant or an entity outside the conference, deciding who can get the floor and when. The floor chair can send its decisions towards the server. The decisions consist of handling instructions, meaning floor accepted, revoked or granted, on one or multiple floors in one floor request. The floor requests are kept in a *floor request set* in floor control protocol level before handling them, while the current floor holders are reflected in *floor holder set*. The items in both sets are not ordered in any priority order, and the sets can be manipulated by the Floor Chair.

The FCS needs to access the centralized conference data, held in form of a *conference object* [Nov07], which contains the floor information, floor controls etc. needed for actually controlling the floors. It is a logical representation of a conference instance at a

[8] The conference object would be typically located in the AS/MFRC, close to its logical users.

certain stage. The conference object is not necessarily located in the same node with FCS but always with the AS/MRFC. This would suggest that the requests to retrieve or to update information will have to be done again using Mp interface when the FCS and the conference object are placed in opposite sides of it.

Low-layer security mechanisms are mandatory for the floor control client and the FCS for floor control relations. Transport Layer Security (TLS) [DiR06] is a mechanism best suited for hop-to-hop security when there is no existing trust relationship between the hosts involved and it needs a connection-oriented transport protocol, in this case TCP, to function. Upon exchanging the first SIP messages, TLS authentication server side is to be agreed by the endpoints involved according to the information provided in the messages. The server is to be one of the conference connection endpoints - the participant or the conferencing system. The endpoint chosen is the one serving as the *answerer* when the mutual understanding for specifics of TCP connection for BFCP was fixed with SDP offer/answer mechanism [Cam06b]. Here the *answerer* denotes the endpoint which receives a session description from another endpoint describing aspects of desired media communication, and then responds to that with its own session description according to the offer/answer model [RoS02]. In practice the authorization is checked in respect to clients and messages. The FCS performs the authority check, even if it were not selected as the TLS-server for the TCP connection. Non-authorized usage of floor control is not permitted and messages are not delivered for further processing – a BFCP status "Unauthorized Operation" is sent as response. The actual usage of TLS is the minimum requirement for BFCP servers and clients [COD06, chap. 7]. Future extensions to BFCP protocol may define additional ways to be supported. In case UDP is used between a conference client and a floor control server, different measures are to be concerned, given in an extension to BFCP [SAH07].

3.3 Protocols in IP Multimedia Subsystem conferencing toolbox

The conference service can be realized using protocols such as Binary Floor Control Protocol (BFCP) [COD06], Session Initiation Protocol (SIP) [Ros02] and Session Description Protocol (SDP) [HJP06]. The distributed media server uses gateway control protocol H.248 between its separated control and user plane. The conference client needs a set of data to manage a connection to a floor control server (FCS): the transport address of the server, the conference identifier, the user identifier and other service and connection related information. These are obtained by using SIP and SDP offer/answer exchange with the conference participant [RoS02, Cam06b].

BFCP does not specify association of a given conference floor and resources, floor creation or floor termination. It needs a call control protocol to be able to function properly. SIP is a natural choice for a companion protocol since it can support the

initiation, modification and termination of media sessions between SIP application server and conference participants to realize the common conferencing operations. SIP is chosen as the session control protocol to be used in IMS and it is heavily promoted for conferencing usage by the various standardization bodies.

Floor control uses SDP to mediate the parameters needed for setting up the floor control client-server connection. Even if SIP is independent of the session description format, the most suitable session description protocol is SDP for actually to carry the BFCP parameters as connection information between a conference participant and floor control server. Moreover, in the IP Multimedia network, the SIP user agents are forced to use SDP as session description format since the Call Server Control Functions (CSCF) must be able to understand the nature of the session. The Mp interface carries the SDP parameters in the H.248 [Itu05] messages to be used in floor control server. The SDP is used in setting up the TCP termination for BFCP usage in case the FCS and TCP connection is terminated in the MRFP. See Figure 3-5 for the protocols involved in the floor-controlled conference session for that network configuration. Note, that according to current 3GPP release 7 specifications the floor control server is in the MRFP, but when the floor control server would be in the MRFC and the BFCP terminated in MRFP, the Mp interface can be used to carry the BFCP messages e.g. tunneled in the H.248 messages.

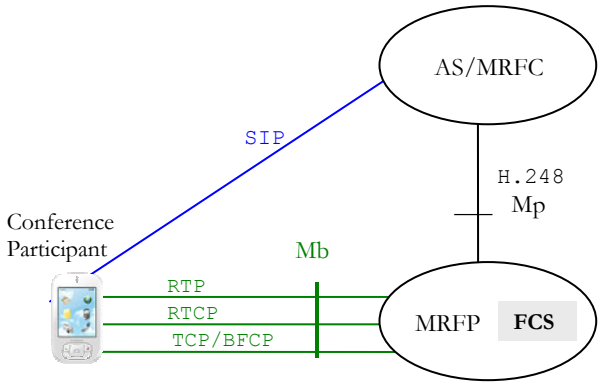

Figure 3-5 The protocols in tightly coupled conferencing in IMS release 7.

For a scenario having the FCS, SIP termination and BFCP termination in the AS/MRFC, the changes caused by floor control involves only volumes and timing of the messages in Mp interface, the changes in protocol parameters are not visible in the thesis analysis level.

The floor control connection data can be resolved also with other means, whereas the SIP/SDP method is chosen as the most feasible alternative. Then the actual connection

for BFCP along with the TCP connection it runs over can be established. The multimedia data as conference payload is transported using Real-time Transport Protocol (RTP) [Sch03], possibly with Real-time Transport Control Protocol (RTCP) [Sch03] for monitoring quality of service (QoS) and to convey information on ongoing RTP session participants, e.g. peer aliveness.

Further in this subsection we present an overview at the two externally observable protocols when looking at the conferencing system, SIP and SDP, to understand the offer/answer mechanism and the new parameters needed for SDP in count of floor control.

Session Initiation Protocol (SIP)

SIP is an application-layer protocol that can be used for creating and managing sessions, which also includes possibility to add and remove different media and participants from an ongoing session. SIP supports five facets of establishing and terminating multimedia communications [Ros02]:

- *User location*: determination of the end system to be used for communication
- *User availability*: determination of the willingness of the called party to engage in communications
- *User capabilities*: determination of the media and media parameters to be used
- *Session setup*: "ringing", establishment of session parameters at both called and calling party
- *Session management*: including transfer and termination of sessions, modifying session parameters, and invoking services.

The formation of Session Initiation Protocol (SIP) dialogs between user agents[9] for two-party connections can be broadened via multiparty conversation sessions to tightly coupled conferencing having a central point of control. This is not included in the main specification RFC 3261 for SIP [Ros02] but basic conference participation is defined in it as complete. More detailed framework for SIP conferencing is depicted in RFC 4353 [Ros06]. Nevertheless, the service for tightly coupled conferencing, studied in the thesis, is not fully realized through SIP conferencing framework. Merely the set-up of the TCP connections to the floor control server, setting up the sessions and tearing them down is assisted by SIP means. SIP in this context deploys the session control protocol role.

[9] SIP user agents are the SIP end systems interacting through an interface most typically, but not necessarily, with a human user. The user agent delivers the media descriptions to the media tools, separate from the agent, acting upon the contents of the description

SIP itself is a merge of two of its ancestors developed in the IETF – Session Invitation Protocol version 1 submitted as a internet draft (SIPv1) and Simple Conference Invitation Protocol (SCIP) [Cam02]. SCIP was designed using Hypertext Transfer Protocol (HTTP) [Fie99] as basis for many of its features and thus SIP inherited some HTTP –like behavior. The current SIP protocol version 2.0 works with both Internet Protocol (IP) version 4 (IPv4) and version 6 (IPv6). SIP is to be used independently of the type of the session and the interworking protocols, although all the SIP elements[10] itself must implement at least TCP and UDP transport protocols.

Request-response model and message bodies. SIP operates per transaction[11] basis. A SIP transaction consists of a request and the response or responses it triggers. A dialog is established after successful responses to an *INVITE* request, and it consists of one or most probably of numerous transactions. A transaction itself can exist outside a dialog e.g. a request for peer capabilities before initial invitation to a session.

IP is a textual protocol and messages comply with basic format introduced in the [Res01]. A SIP request and the recipient are identified by the first message line, the *request-line*. It states the type of the request – the *method*, a Uniform Resource Locator (URI) [Ber05] in form of a *request-URI* and the SIP protocol version. The more accurate usage and Backus-Naur Form (BNF) of the general SIP header are found in the [Ros02]. An example of a SIP request with minimum required *headers* for it and a session description, as a single *SIP body*, according to Session Description Protocol (SDP) [HJP06], seen here after the blank line, starting with "`v=0`":

```
INVITE sip:signal@ericsson.com SIP/2.0
Via: SIP/2.0/UDP wall.ericsson.com;branch=z9hG4bK776asdhds
Max-Forwards: signal   To: Benita <sip:signal@ericsson.com>
From: Aila <sip:aila@ericsson.com>;tag=1928301774
Call-ID: a84b4c76e66710@wall.esignaln.com
Cseq: 322344 INVITE
Contact: <sip:aila@wall.ericsson.com>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 wall.ericsson.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

10 A SIP element is a user agent client or server, stateless or stateful proxy or a registrar.
11 The transaction should not be confused with a *dialog* which is the peer-to-peer relationship sharing the same dialog identity.

The request-URI indicates the address of a user or service the request is for, and in SIP is one of the following: general URI, "sip" URI [Ros02, subsection 19.1], "sips" URI [Ros02, subsection 19.1], or, according to additional support at the user agents, probably another URI scheme.

Furthermore, a request contains zero or more SIP bodies and some lines called *headers* [see appendix A.2 for table of all SIP headers] giving information about the request, session, and the bodies the message holds. A SIP body typically contains a session description as the payload, not processed by the proxies.

Methods, responses and their usage. There are six basic request types, methods, defined: *INVITE*[12], *ACK, BYE, CANCEL, OPTIONS* and *REGISTER*. Additional methods are defined in extensions to the core SIP. Figure 3-6 gives an example of a normal SIP setup and release dialogue, the entities could be, e.g., Alice as User Agent Client (UAC) and Bob as User Agent Server (UAS):

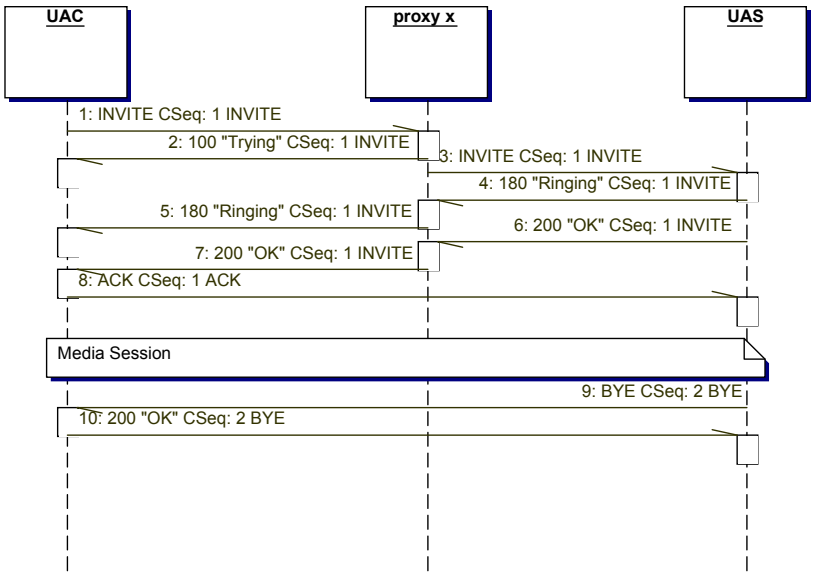

Figure 3-6 A Typical SIP dialogue with one provisional response (5)

An (initial) *INVITE* request initiates the session between the peers, while a *BYE* request terminates the session or attempted session between peers. Also the dialog associated with that session is terminated with it. A user agent may query information about a proxy or another user agent by using an *OPTIONS* request. To use location

[12] The method names are historically spelled with capital letters.

service, the proxy server or a user agent reads and updates the contents of the SIP registrar mappings using a *REGISTER* request.

The methods in SIP mainly use two-way handshake – a request from the client has to get a reply in form of a *final response* from the server to finish the transaction. An exception of the two is the *INVITE* method, which uses *three-way handshake*. It means in SIP that after the final response is received to the request as in normal two-way handshake, the client further replies with an *ACK* request. The *ACK* is used only for this purpose and it does not trigger any responses. The approach taken when designing the core of SIP was to put weight on whether the session was successfully established or not. Following the setup was left to be decided by the implementers [Ros02]; for this reason, not all responses are final, there are also *provisional* responses that do not end the transaction, but do provide information of the call telling that the request is being processed. Provisional responses can be such as "Session Progress", "Call Is Being Forwarded" or "Ringing", see appendix [A.3] for table of all SIP response codes. These responses can be very useful e.g. in a cellular network and SIP interworking Voice over IP (VoIP) solution, where the mobile caller side has to be informed of the ongoing time-consuming paging[13] of the called subscriber over the mobile realm.

Modification of terms. While the dialogue is still on and the initial *INVITE* transaction is concluded, the parties can change the terms of the session or dialog by supplying a new session description. The user agent sends the new session description in a *re-INVITE* within the existing dialogue. Another way to modify the dialogue is to use message headers to change the remote target.

Reliability in transmission and congestion control. The SIP elements must support both TCP and UDP, but retransmissions and timing of messages in user agent core level is to be applied only when using unreliable transports. The details of retransmission and timing are not crucial for aims of BFCP; they are discussed in larger extent in SIP specification [Ros02] and hence are not touched here further. SIP does not offer any congestion control by default. The protocol does not take into consideration even cases where there is media failure, but does give recommendations or mandates to use different transport protocol. The specifications suggest, that non-congesting measures would be a good idea to deploy, e.g. trying to avoid automated generation of messages exhaustingly as trying to end the session or to the keep it alive with *re-INVITEs*.

Services and extending the protocol. The standard includes descriptions on how to widen the scope of the protocol. That is accomplished by using the *SIP Extension*s

[13] The search for a cellular user in the network in the place, where the network has informed the user equipment is located, is called paging.

[Ros02, RoS06, Cam02]. The SIP messages are routed mostly in the same way as mail messages; therefore they can also carry multipart message bodies [FrB96] to ease the service implementation. The only services SIP offers are security related.

Session Description Protocol (SDP)

SDP is a widely applied session description format. As it does not have transport as a feature, it uses carriers, such as SIP, Hypertext Transport Protocol (HTTP) [Fie99], Real Time Streaming Protocol (RTSP) [SRL98], Session Announcement Protocol (SAP) [HPW00] or a protocol using Multipurpose Internet Mail Extensions (MIME) [FrB96]. The SDP session description is indicated in deployed protocols by the media type "application/sdp". Traditionally the SDP session description instances are called shortly as *SDP*:s, lining up with the protocol name.

An SDP must express satisfactory set of information to consent applications to join a particular session or to announce the existence of a session and resources in use or required of an application. The SDP includes information on the session itself, media used and timing factors. SDP session description, typically as a payload in a SIP body, contains peer information e.g. IP addresses, port numbers, the multimedia type to be transported along with the protocol type, e.g. "TCP/TLS/BFCP". Also categorization of the session descriptions to segregate between interesting and irrelevant description instances is possible using SDP attributes. Security aspects are irrelevant from SDP perspective and are typically enhanced by encrypting the session description, but a cryptographically sufficient encryption key exchange mechanism is not in scope of the core SDP [HJP06]. The simple mechanism provided is not recommended for use, whereas the key management should be done with assistance of a SDP extension [Ark06][ABW06] or by additional means if necessary for the session. The SDP may also have URIs to instruct for obtaining auxiliary data needed for deciding about the session specifics.

Syntax. The SDP session descriptions are fully textual to enhance portability, the protocol recommending a specific text encoding but allowing also multiple other choices. The rather compact encoding, strict order and formatting of the lines the SDP consists of assists in parsing the description and in spotting misfit instances or encrypted session announcements with no accurate interpretation available. The separate lines are formed as

```
<type>=<value>
```

where the *type* is one letter denoting the type definition and the *value* field being by

default case sensitive structured text, construction of which is fully dependent of the type. An accurate line could be for example "`m=application 9 TCP/TLS/BFCP *`", meaning the media at hand is an application using the protocols TCP, TLS and BFCP. Below is an example of a simple SDP description [HJP06] with valid mandatory fields:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

The SDP always consists of *session-level* lines including also timing information forming *time descriptions*. The *media-level* lines are not mandatory but when present, they form one or multiple *media descriptions*. As mentioned earlier, the format of the SDP is authoritarian giving the order of the lines as pre-defined. Part of the lines are mandatory, rest of the lines are optional. The session description, time description and media description order with explanations of the individual lines are given in Figure 3-7, optional lines marked with a "*".

```
Session description
   v=  (protocol version)
   o=  (originator and session identifier)
   s=  (session name)
   i=* (session information)
   u=* (URI of description)
   e=* (email address)
   p=* (phone number)
   c=* (connection information - not required if included in
        all media)
   b=* (zero or more bandwidth information lines)
One or more time descriptions ("t=" and "r=" lines; see below)
   z=* (time zone adjustments)
   k=* (encryption key)
   a=* ("er" or m"re"session attribute lines)
   Zero or more media descriptions

Time description
   t=  (time the session is active)
   r=* (zero or more repeat times)

Media description, if present
   m=  (media name and transport address)
   i=* (media title)
   c=* (connection information - optional if included at
        session level)
   b=* (zero or more bandwidth information lines)
   k=* (encryption key)
   a=* (zero or more media attribute lines)
```

Figure 3-7 SDP types and order [HJP06]

The mandatory types *v*, *o*, *s*, *t*, and *m* are introduced briefly in the next paragraphs, other types can be found in the core SDP specification or in the Augmented Backus-Naur Form (ABNF) grammar for SDP [appendix A.1].

The first type ***v*** denotes the SDP protocol version for the description, which is 0 (zero) being the only and currently the latest version defined.

The originator of the session is given by the type ***o***. The value is constructed as

```
o=<username> <sess-id> <sess-version> <nettype> <addrtype>
<unicast-address>
```

The session version is typically a time stamp indicating a version upgraded with modification to the session data. The rest of the value sub-fields builds a globally unique identifier for the session. The tuple comprises of originating user identification, originator dependant session identifier, network type of a registered type ("IN"=internet), a registered type of the address following and the address of the originator. The simple value "-" is used instead of the user identification if not available.

The ***s*** field denotes the session name. It can not be omitted but can be substituted with a single space character when the field is not meaningful.

Timing of the session is handled with the ***t*** field. A single line indicates the beginning and end of the active session. The session can be set as permanent when the boundaries indicate zero. The session can be repeated using various times, multiple lines are used for this purpose. For a session that is not bounded by ending time, the assumption for active session duration is half an hour.

Media descriptions start with the ***m*** type. The field is divided into subfields as

```
          m=<media> <port> <proto> <fmt> …
or        m=<media> <port>/<number of ports> <proto> <fmt> …
```

The *media* gives the media type, which currently comprises types "audio", "video", "text", "application" and "message". The *port* is the port where the media can be received. If multiple ports and multiple addresses (in ***c***-line) are defined, a one-to-one mapping is assumed. The *proto* denotes the transport protocol where the meaning is dependent on the ***c***-field and it usually has the value "RTP/AVP" pointing out to audio and video profile of Real-time Transmission Protocol (RTP) with minimal control running over UDP. The *fmt* subfield or subfields are giving the media format descriptions dependent on the *proto* subfield.

SDP Attributes. The **a**-field is the mechanism to extend the definition of the SDP. The field can appear on media or session level or in both simultaneously. The two formats

```
          a=<attribute>
or        a=<attribute>:<value>
```

indicate the owner of the attribute. The format without a value is to be considered as property of the session, while the latter format conveys that the attribute should have a desired value, e.g. "a=sendonly" or "a=charset:ISO-8859-1".

SDP offer/answer model. The SDP specification lacks instructions on how the exchange of the SDP instances between the peers is performed. The nature of SDP can be described by a quote: "Note, that although the *P* in the *SDP* stands for *Protocol*, SDP is simply a textual format to describe multimedia sessions" [CaG05].

The offer/answer model has two agents as participants: the *offerer* and the *answerer*. The offerer generates a SDP of its own preferences, called an *offer*, and delivers it to the answerer. The answerer then sends back an *answer*, which includes matching media stream lines for exactly the ones in the offer. The answer has to indicate, for each stream, whether it is accepted or not and what are the equivalents for the codecs, ports and addresses in the answerer end. When an agent wishes to send or receive several media streams at the same time, the offer or answer has to include all streams.

The SDP offer can always be rejected or approved, and it can also be generated at any time. The SDP rejection causes the session to fall back to the state previous to the offer regardless of the state itself – whether the session existed or not. The exceptions to those are situations, where an offer is received but yet not processed thoroughly, or an answer (or rejection) is waited for an offer sent. They might put the new information on queue to wait for the existing processing to end.

The same offer/answer mechanism is used for updating the session information after the first, *initial offering*, performed outside any context formed with a higher layer protocol (in this case SIP) between the two agents. The non-initial offer can be same as the initial offer or different. The answer can also be the same or different as the offer it replies to. The offer/answer mechanism, defined as mandatory baseline for SIP interworking, is described as a whole in *RFC 3264* [RoS02].

SDP for TCP or TCP/TLS. The core SDP specifies three transport protocols: "udp", "RTP/AVP" and "RTP/SAVP", all of which are running on UDP – the simple "udp" denoting an unspecified protocol on UDP. The core BFCP is running on top of TCP

or in addition also on TLS. The new transports "TCP/BFCP" and "TCP/TLS/BFCP" [Cam06] need new transport definitions for the SDP as a base: "TCP" [YoC05] and "TCP/TLS" [Len06].

The "TCP" protocol identifier, aligned with the "udp" identifier, does not specify the upper layer protocol and it is further specified in the `fmt` subfield of the same `m`-line. The "TCP" identifier definition is accompanied with two SDP attributes – "setup" in the media or session level and "connection" in media level:

```
a=setup:<role>
a=connection:<conn-value>
```

The setup connection roles indicate, whether the endpoint wants to initiate the TCP connection ("active"), to accept an incoming connection ("passive",), do both of the former ("actpass") or to hold the connecting for the time being ("holdconn"). Updates for the connection attribute of SDP shows, whether the endpoint still wishes to continue using the existing TCP connection ("existing") or to make a new one ("new"). For example, if the endpoint has noticed that the TCP connection for a media stream is closed, it should make a new offer with connection value "new" for that stream.

The confidentiality, integrity, and authenticity are values appreciated for multimedia sessions. The endpoints often do not have possibilities to obtain sufficiently signed authentication certificates, and self-signed certificates are usually the only option. When the integrity and confidentiality of the SDP is secured, the SDP can include a *certificate fingerprint* – a secure hash of a certificate. Those can be securely used further for verification in the TLS handshake to introduce trust relationship in spite of usage of self-signed certificates. For this purpose, the "TCP/TLS" identifier definition brings along the `fingerprint` attribute, used both in media and in session level:

```
a=fingerprint:<hash-func> <fingerprint>
```

The subfields denote the hash function used for the fingerprint and the fingerprint itself. An example of the attribute can be seen in Figure 3-9 in *SIP INVITE* and *SIP 200OK* messages. The details of the subfields are described in the specification and ABNF for the attribute [Len06].

3.4 Common conferencing operations using Session Initiation Protocol

The common conferencing operations performed with SIP in the IMS include creating, joining and tearing down a conference, removing and obtaining information on participants and manipulating the media graph. The descriptions in this subsection are given in protocol method level. All call signaling is subject to policy control and is affected by the consultation to the conference policy, also during set-up and release phases.

The conference creation takes place using ad-hoc methods, which means that the conference is unscheduled and is created on-the-fly by a conference participant. The SIP user agent issues a standard SIP *INVITE*, including the session description reflecting the request for floor-controlled conference connection. A successful call attempt towards the conferencing system creates a new conference and a focus with an identifying conference URI. When a conference factory (used for conference URI creation) is utilized, the call can be directed straight to it which will create the conference and the focus automatically. An example of a participant using SIP call signaling for starting the conference is shown in Figure 3-8. Floor control can be deployed between the participant and the floor control server (FCS) when the TCP connection is up. The live conference floor manipulation is done with BFCP over TCP.

The conference URI must be provided for the new participants when they come to the conference by first-party addition. The *third party addition* is done by a participant, requesting the new user with a *REFER* to send an *INVITE* to the conferencing system, or, the conference member participant sends a SIP *REFER* to the conference focus. It will initiate a SIP dialog towards the new user by sending an *INVITE*. The benefit in the second approach is that it allows the users not supporting the *REFER* method to still join the conference by SIP means.

The media graph of a conference can be influenced by adding and removing media streams or manipulating the stream modes i.e., removing or changing the participants' rights to receive and send a stream. One way to do these is via SIP re-*INVITE*. A participant can send a re-*INVITE* request with an altered session description to the focus. This is referred as *first party signaling*, and it does not affect states of other parties in the conference. In case the focus needs to alter the media graph of participants, it will send them re-*INVITEs* with new session descriptions. When receiving the new media description, the participant has to choose whether to accept the change or not. Another way to change the media graph is an initiative from conference system side to change the stream modes by just ordering the change in Multimedia Resource Function Processor (MRFP). Binary Floor Control Protocol specification does not actively

support altering the media graph but gives information on floors in such a way that the floor control server is capable of carrying out the actions needed to accomplish media modifications if the conference policy requires it.

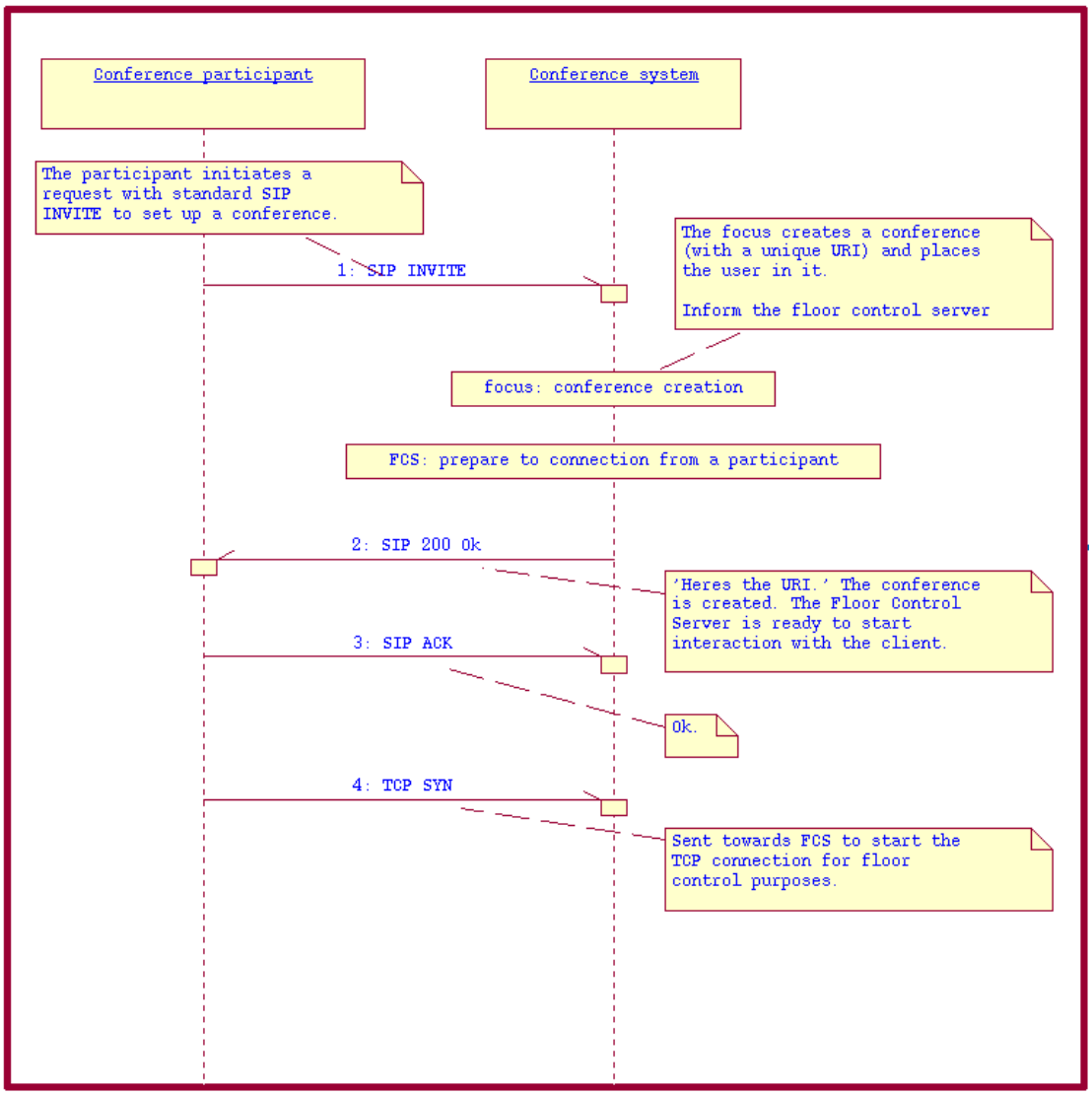

Figure 3-8 The participant creating a conference.

Acquiring information on a conference or participants joining or leaving the conference can be done by deploying the notification service. The service takes subscriptions, according to which it notifies the requesters of intended or concluded actions. The notification service can be realized using SIP-specific event notification [Roa02] and SIP event package for conference state [RSL06]. The event notification service is outside of the scope of the thesis because it does not involve floor control and it must not be confused with the floor control protocol information procedures partially overlapping functionality.

A release of a conference or a participant can be initiated by a focus (possibly with influence from the conference policy) or a participant. The conference or connection tear-up is done using the standard SIP procedures for call release: a standard *BYE* is issued to release a call signaling relationship between two participants - the procedure is called *first-party tear-up*. The *third-party* departures are the tear-ups, accomplished using the *REFER* method. The conference URI, policy and conference data are canonically destructed when the conference is released. The most common conference release takes place, when all or a specific participant has left the conference [14].

3.5 Setting up the floor control server connection

We clarify in this subsection the usage of SDP with SIP in assisting floor control for conferencing scenarios. The FCS connection establishment outside the SIP offer/answer model is described in RFC 5018 [Cam07] but is not considered in the current scenarios nor in the thesis.

Adjusting SDP. The BFCP connection is described as a media stream running over TCP (using possibly also TLS)[15]. It is done with help of the SDP media description (media field) and some standard and new SDP attributes [Cam06b]. The new lines and needed values with their usage are introduced in the following paragraphs.

For each floor-controlled media stream the `media` field (`m` line) in a SDP description is to be set to value "application" and the `transport` subfield for core solution can be either "TCP/BFCP" or "TCP/TLS/BFCP" indicating the floor control protocol stack order . The port to be used in the subfield `port` is naturally a TCP port, value for which is to follow the rules in RFC 4145 [YoC05] and RFC 4583 [Cam06b]. The format subparameters for the line are omitted and should be covered by a single "*". An example of a valid m line could be e.g. `m=application 50000 TCP/TLS/BFCP *` for a connection using TLS.

The roles for the endpoints are determined per BFCP connection basis. The configuration is seemingly simple; there are only one-to-one server-client relationships. This does not exclude situations where both endpoints act as floor control servers, but for different streams. The SDP media-level attribute for the role determination, support of which is mandatory for the endpoints wishing to use floor control, is

```
a=floorcontrol:<role> …
```

[14] This can be ruled by conference policy
[15] BFCP allows one TCP connection per client, identified by a distinct user ID.

The attribute denotes the role of the endpoint in question and can indicate also multiple roles. The choices are client, server or indication of willingness to act as both. The default role, when the attribute is omitted in the offer, is client, although the answerer is strained to use the attribute whenever the offerer is using it in the offer.

A conference, a participant and a conference floor must be identified externally. Three media-level attributes are addressed to this need and are to be supported by the endpoints:

```
        a=confid:<id>
        a=userid:<id>
        a=floorid:<id>
or      a=floorid:<id> mstrm:<id-ptr> …
```

The conference and user identifiers are integer tokens that are provided and normally included in the session descriptions by a floor control server. A specific floor is associated with streams using the *floorid* attribute, where the stream pointer as subfield is formed using *label* attribute as described in RFC 4574 [LeC06].

The TCP or the secured TLS over TCP connection setup and reestablishment is managed using the *setup* and *connection* attributes in the way described earlier. The reasons for reestablishment are given in the RFC 4582 [COD06] and the procedure for it in RFC 4583 [Cam06b].

SIP usage for floor-controlled conference. SIP can be used to build up services with help of the collection of primitives it already provides. Most of the SIP servers in the network ignore the contents of the session description the message carries, therefore the description protocol of the body, usually SDP [HJP06], can be targeted or even changed to one better suited for the purpose without changes to the mediating network itself. The basic SIP functionalities and primitives suffice for linking a participant with the FCS and in usage of floor control for a conference.

The user wishing to set up a conference sends an initial SIP invitation to the conference application server. This is a standard invitation with the difference, that it uses in the offer/answer exchange the exact SDP attributes and values brought up in this subsection to signify the intention to contact FCS. Whenever the application server has consulted the conference policy, it allows the set-up to continue by sending an answer in form of SIP *200 OK* to the user. When the SIP offer/answer exchange is performed, the parameters extracted from the approved SDP are propagated to the FCS. The received SIP *ACK* request indicates for the user that the conference is created and there is a guaranteed access to the FCS. The endpoint decided to be the

initiator of the TCP connection, in this case the conference participant, then continues the interworking with TCP *SYN* primitive (including the TSL procedures if applied).

The conference participants get knowledge of the conference by invitation from conference system or, e.g., via voice- or web applications. Whenever they get the first contact with the conferencing system, the standard SIP offer/answer mechanism is deployed in the way analogous to the one described for the first participant. An example of a FCS connection set-up with SIP and SDP usage is visible in Figure 3-9. In the figure the SIP messages are edited in some extent; SIP headers not essential to the FCS connection are not shown.

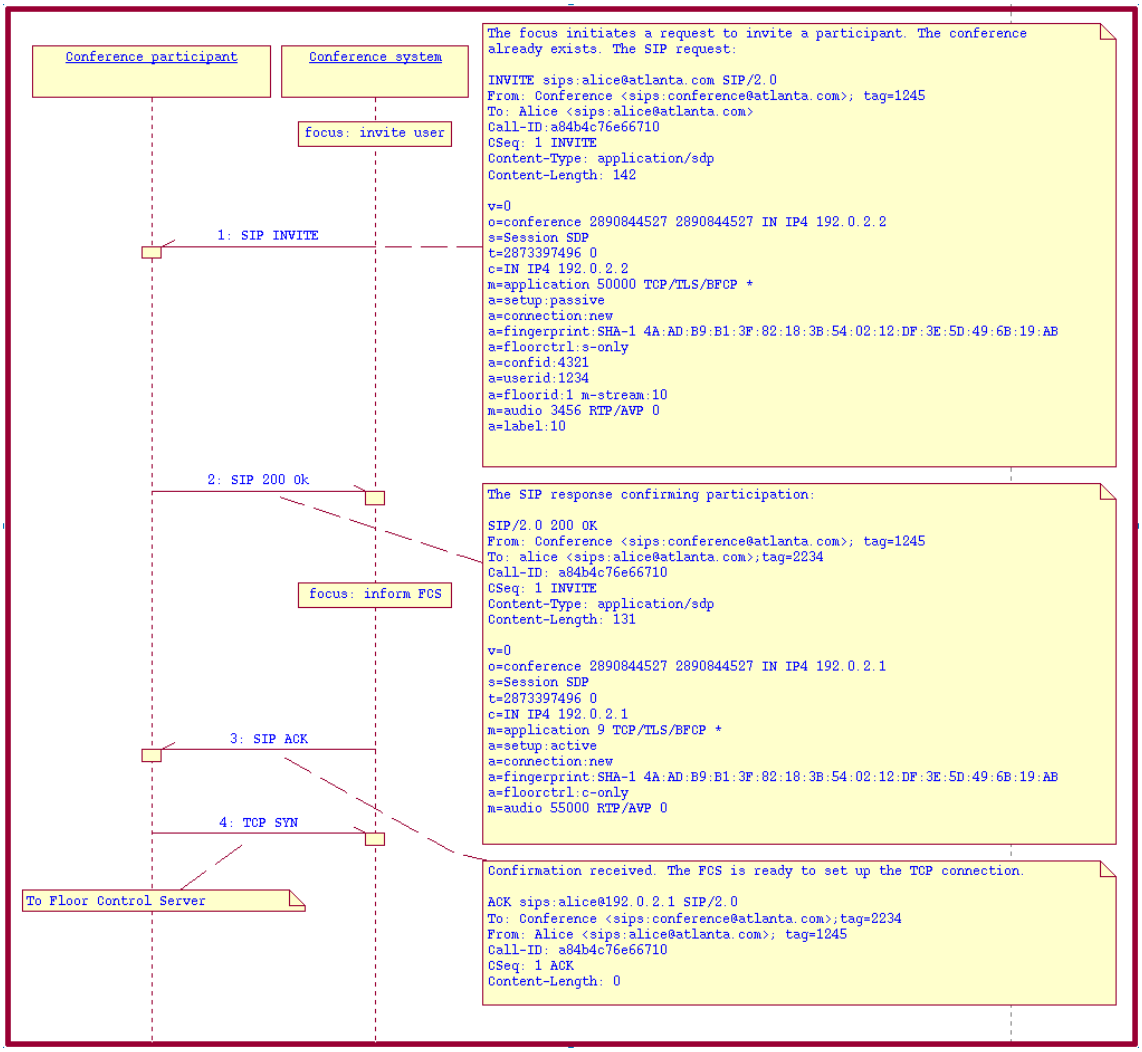

Figure 3-9 The focus inviting a participant [Cam06a]

Each time any legitimate changes to the media are performed from conference participant's behalf e.g. by adding or removing a media, the SIP interface endpoint in charge of the change may send a SIP *re-INVITE* with a new SDP offer towards the

other endpoint. This is done according to the SIP/SDP offer/answer exchange, but it is a choice in implementation whether some of the media changes during a live conference are carried out by the decision of the media server or application server alone without consulting the conference participant. Furthermore, even if the floor control will finally discharge the conference resources, the release of the session relationship is performed from either end using a standard SIP *BYE* method.

4 Protocol and mechanisms for floor control

In this section we examine the Binary Floor Control Protocol (BFCP) for tightly-coupled multiparty conferencing. Floor control is a combination of mechanisms that enables entities to divide shared conference resources in mutual understanding. BFCP is defined for this task to enable coordination of access to floors. The purpose is to enlighten the nature of the protocol at hand and to provide the reader with basic understanding on protocol mechanisms, described according to the IETF specifications. The needed attribute and primitive summaries are provided in appendices of the thesis, the rest of the details can be found in BFCP specification [COD06].

4.1 Overview

BFCP is a novelty in telecom and internet worlds. Various protocols have been specified for different models providing floor-control-like services in distributed conferencing. Still, none of the floor control protocols has been quite convincing for in flexibility, openness and usability until BFCP was introduced. The BFCP protocol and its features, as much as has been covered until now, are defined in a fresh set of IETF Requests for Comments (RFC) and drafts. The requirements for the protocol are set in RFC 4376 [Kos06], connection establishment performed without the offer/answer model is described in RFC 5018 [Cam07] and the actual protocol is defined in RFC 4582 [COD06]. BFCP runs only over TCP, although an extension [SAH07] enabling it to run over UDP is available. The extension introduces some changes to the transaction model, e.g., all messages have a response, which will delay the transactions to some extent. The usage of UDP as transport is not separately considered in this thesis.

BFCP protocol uses binary encoding. This results in smaller message size that helps to cope with incidents of low-bandwidth and transferring the delay-sensitive messages as opposed to textual protocols. The delay-sensitive BFCP messages are not expected to grow in size with potential protocol extensions in the future.

The floor control protocol is meant for passing the floor control messages between the floor chairs, the floor control server and the participants involved during a conference, its set-up and tear-down. The architecture explained earlier is assumed, provided that the possibility on existence of multiple floor control servers or MRFPs involved in a conference is not taken into account in this section.

The concrete floor creation, obtaining floor resource associations or information to contact a floor control server and floor control privileges are not in the scope of BFCP but are essential for the operation of the protocol. These are discussed in more detail in the framework for centralized conferencing [BBL07]. BFCP can be used also as a stand-alone protocol with SIP without a conference policy control protocol. The policy control would be used for authorization and limitation of access to conference resources depending on predefined rules by the network operator or conference arranger.

4.2 Transport

The main reasons why TCP is used to carry the BFCP messages is due to reliability in delivery of a stream of data between its users. The fact that the carried packages containing segments of the original data should be expected to be delivered in order was also a major shaping factor when the protocol stack was constructed.

TCP connection is set up per client basis; here a client is a BFCP entity using a distinctive identity. When the TCP connection is not able to deliver BFCP messages or an entity receives corrupted data which is impossible to be parsed, the connection is timed out, closed and reestablished. The reconnection is done similarly to the initial connection to FCS. While the floor participant is behind a dead TCP connection, the pending requests should be held in reserve for the time of the reestablishment, but this is, after all, subject to local policy. A graceful TCP connection close can be kept as an indication from a floor control server, on behalf of the focus, or indication from a participant to wish to end the floor control relationship.

Since TCP takes responsibility only of the accurate order of the message data flow, BFCP employs message framing at the application level – the packets are binary encoded using *type-length-value* (TLV) elements. This is applied inside the messages in the 12 bytes (here equals as octets) long *common header* and in the *attributes* trailing it. The elements consist of the element type in numeric code, length of the value and a variable byte-sized value. The length of those first two fields in a TLV coded element is fixed in bytes. This makes parsing of the messages rather fast and skipping unwanted elements safer.

4.3 Packet format

There are two types of attributes – TLV encoded and group attributes, both types following a common header in the BFCP message. The summary of message, common header and attribute formats are given in the next paragraphs. The summary of formats of the attributes and messages as a whole is given in the appendix [A.4, A.5] of the thesis; the detailed descriptions on packet formats can be found in the BFCP specification [COD06].

COMMON-HEADER format. The common header for BFCP version 1 is used for all messages and is of format:

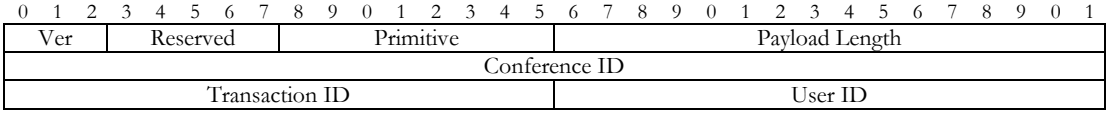

Figure 4-1 BFCP *COMMON-HEADER* format

The *Ver* field is set to 1 to indicate the protocol version. The *Primitive* field indicates message type, the *Transaction ID* matches the requests and responses, while the *User ID* exclusively identifies a user within a conference. Generally the *User ID* field is mapped to the identification used in the session establishment protocol; in this case the session is established with SIP and the field matches the SIP *Request-URI*.

Attribute format. All of the attributes follow a general format:

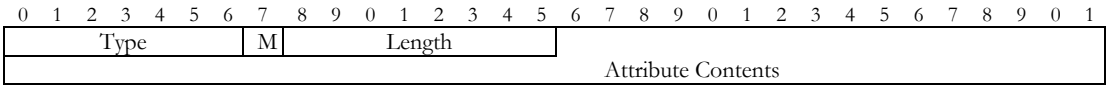

Figure 4-2 BFCP attributes common format

The *Length* gives the size of the attribute contents, whereas the attribute contents itself vary. The four attribute formats are 16-bit unsigned integer, 16-bit arbitrary data (octet string), arbitrary data (octet string) of variable length or *Grouped* – a sequence of attributes. The *Mandatory –bit* (*M-bit*) serves a special purpose; it indicates whether the entire message is to be rejected if this specific attribute is not recognized. The core protocol attributes are all considered as mandatory, and hence the bit is meaningless regarding support of the attributes enlisted in the BFCP main specification [COD06]. There are 18 attribute types, listed here with their formats for the contents:

Type	Attribute	Format
1	BENEFICIARY-ID	Unsigned16
2	FLOOR-ID	Unsigned16
3	FLOOR-REQUEST-ID	Unsigned16
4	PRIORITY	OctetString16
5	REQUEST-STATUS	OctetString16
6	ERROR-CODE	OctetString
7	ERROR-INFO	OctetString
8	PARTICIPANT-PROVIDED-INFO	OctetString
9	STATUS-INFO	OctetString
10	SUPPORTED-ATTRIBUTES	OctetString
11	SUPPORTED-PRIMITIVES	OctetString
12	USER-DISPLAY-NAME	OctetString
13	USER-URI	OctetString
14	BENEFICIARY-INFORMATION	Grouped
15	FLOOR-REQUEST-INFORMATION	Grouped
16	REQUESTED-BY-INFORMATION	Grouped
17	FLOOR-REQUEST-STATUS	Grouped
18	OVERALL-REQUEST-STATUS	Grouped

Table 4-1 BFCP attributes

Messages. The ABNF form of the messages is found in the appendix [A.5]. The defined 13 messages are listed in the table below with a short explanatory text and allowed message directions:

Value	*Primitive*	*Description*	*Direction* *S – FCS* *P – Participant* *Ch – Floor Chair*
1	FloorRequest	Request a floor	**P → S**
2	FloorRelease	Release a floor or a pending floor request	**P → S**
3	FloorRequestQuery	Inquire information on floor request	**P → S ; Ch → S**
4	FloorRequestStatus	Inform of status of a floor request	**P ← S ; Ch ← S**
5	UserQuery	Inquire information on users and their floor requests	**P → S ; Ch → S**
6	UserStatus	Inform of participants and their floor requests	**P ← S ; Ch ← S**
7	FloorQuery	Inquire information on floors	**P → S ; Ch → S**
8	FloorStatus	Inform of floors	**P ← S ; Ch ← S**
9	ChairAction	The chair instructs the server	**Ch → S**
10	ChairActionAck	The server has accepted the ChairAction message	**Ch ← S**
11	Hello	Check the liveliness of the server	**P → S ; Ch → S**
12	HelloAck	The server is alive and accepted the Hello message	**P ← S ; Ch ← S**
13	Error	Errors in processing requests	**P ← S ; Ch ← S**

Table 4-2 BFCP primitives

4.4 Operation

BFCP protocol defines three roles for entities: the floor participant, the floor chair and the floor control server. In Figure 4-3 the logical entities are depicted along with the primitives provided by the protocol.

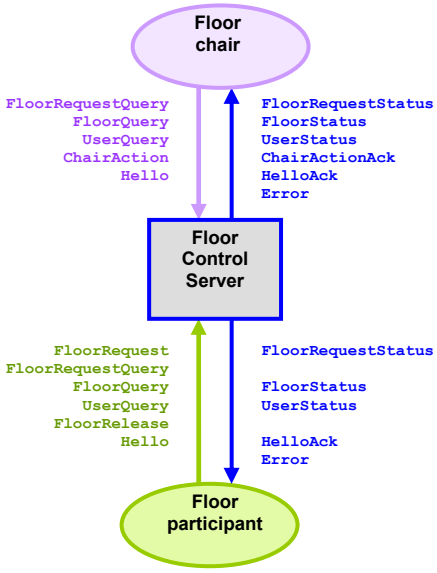

Figure 4-3 Primitives provided by BFCP

The BFCP transactions are of two sorts: client initiated, which consist of one request with one response to it, and server initiated transactions with no matching response. The messages belonging to a transaction are recognized from identical Transaction IDs in the message header. Since the server initiated transactions do not have a reply, they are called simply notifications and the lack of need for coupling is indicated with the Transaction ID as zero. The notifications are to keep participants and chairs informed about the status of the floor or the floor request submitted.

There are seven entity transaction procedures in core BFCP. The procedures are *Floor request procedure, Floor cancelling or floor releasing procedure, Chair action procedure, Floor information procedure, Floor request information procedure, User information procedure,* and *Server capability procedure*. Information on floors, floor requests and users can be obtained using BFCP methods and these procedures or with additional means outside the scope of the BFCP. The procedures described all involve floor control with core BFCP and are explained further in the following paragraphs.

Floor request procedure. When the conference participant needs a permission to use a floor, it can be acquired from the FCS with a *FloorRequest* message. The participant can have many floor requests pending on the request queue. The server tries to answer with a *FloorRequestStatus* as soon as possible indicating the status [appendix A.5.4] of the request. The response or an *Error* message concludes the transaction, even if the request was not accepted or denied yet. State changes trigger the server to send subsequent *FloorRequestStatus* messages to the floor requester until the floor or floors are granted. The floor control server may choose to create the status messages only on selected occasions, e.g. when the actual floor request status changes but not on advances in the request queue. The floor request is considered as alive when it has not

reached state "Cancelled", "Released" or "Revoked", but whenever it does the floor control server can discard the relating state information.

Floor cancelling or floor releasing procedure. A user can remove an ongoing floor request or remove it from the floor sets from the server by sending a *FloorRelease* message to the server. As both of the occasions logically stand for the wish to release a specific floor or floors, the actions are handled by the same procedure. The response ending the transaction can be either *FloorRequestStatus* or *Error*.

Chair action procedure. A floor chair can instruct the server to grant or revoke a floor. The *ChairAction* message is placed to this purpose and can be applied simultaneously to many floors within single floor request. The floors denoted in one floor request are divided in the server to different pending queues (and messages) for the chairs corresponding the floors in the request. The floor control server will combine the *ChairAction* messages belonging to a single floor request – the floor request is still handled as a whole which means that if one of the floors is not granted, all of them are denied. The floor chair may include in the floor request the reason of the decision or the queue holding in the message. Note that if the floor control server implements the queues, they can be exclusively for a floor, common for all floors or a variation with both. When the *ChairAction* message is received at the server, it will act upon the indication of the message. The *ChairActionAck* as a response to the chair means that the server has understood the action, but the server still may linger on fulfilling the task as it is responsible to keep a coherent state and acts only when its own state allows it. An *Error* message instead of the *ChairActionAck* as a response is also considered as end of the transaction.

Floor information procedure. Information on floors status can be obtained by sending *FloorQuery* message to the floor control server. The status of a floor consists of information on all the floor requests for the specified floor or floors and can be dependent on the conference role of the requester. The server replies first as soon as possible with *FloorStatus* or *Error* to end the transaction, then periodically informing on changes on the floor with a *FloorStatus* message. This is continued until the requester seizes the procedure for all or specific floors with a new *FloorQuery* with the respective *FLOOR-ID*s omitted. Since one *FloorQuery* point to many floors, several *FloorStatus* messages (as one is having information of a single floor) may be needed to carry out the task.

Floor request information procedure. The floor request status, including information on all floors on a request and their advances in the floor request queue, can be asked from the floor control server with a *FloorRequestQuery* message. The response to it, ending the transaction, will be a *FloorRequestStatus* or *Error*. The

mechanism works in a similar way as for the *Floor information query* but with two deviations. Firstly, the response can carry information on multiple items – here on floor requests. Secondly, the procedure has such an addition that the *FloorRequestQuery* message can be used also in polling quality, e.g. to fill in for cases where the server chooses to reply with the *FloorRequestStatus* to *FloorRequest* only for some specific changes for a floor request, but not all.

User information procedure. The client can ask for information on a specific participant and the relating floor requests with a *UserQuery* message towards the floor control server. The information delivered with *UserStatus* can be useful e.g. after TCP reconnection to find out about the floor requests that are still alive or to find out user URI etc. The *UserStatus*, as also the *Error* message instead it, ends the transaction and procedure.

Server capability information procedure. In addition to the "ping" –like functionality, the *Hello* message serves as a server capability information query means. The clients can use it to acquire information on primitives and attributes supported by the server. The response *HelloAck* or *Error* will end the transaction.

The simple example in Figure 4-4 shows a typical conference set-up from user side. The set-up from a single user and FCS perspective with *FloorRequest* and *FloorRelease* procedures are shown with usage of characteristic attributes. The figure does not show other participants of the conference, but as long as the conference is up and running and the *Conference ID* is obtainable, any user allowed to take part with a legitimate TCP/BFCP connection may start with the *FloorRequest* procedure.

The signaling sequence starts with the participant initiating a session with the conference system as described earlier. When the TCP connection towards FCS is established, the participant requests for a floor from the fresh conference. The server first accepts the *FloorRequest* message and replies the participant that the message is pending for processing. The participant is informed of *FloorRequest* status changing twice more: first it is accepted and put to queue and then the floor is granted. After a while the participant decides to release the floor and does so with a *FloorRelease* message. The server replies to acknowledge the release.

The conference policy will have its saying when the conference is released, but a usual case would be, that the participants quit first from floors with BFCP *Floor cancelling or floor releasing procedure* and then with a graceful TCP close and a SIP *BYE*.

4.5 Security

The mutual authentication between clients and servers, replay and integrity protection and confidentiality are the reasons why TLS is used for BFCP. The *non-null encryption* manner or other mechanisms providing similar security properties are recommended to be used with BFCP.

The conference participants and floor control servers are recommended to be authenticated at initial contact. All the floor control messages should be authenticated and integrity-protected when received or sent.

4.6 Extending the protocol

BFCP is built future-proof and scalability in mind. The protocol core does not include an exhausting set of definitions for different kind of purposes but all the principles of operation and syntax to form a capable, flexible protocol. Even if the standard defines a complete and usable protocol, it also is possible to widen the scope of the protocol. This is accomplished by using *extensions* that can be designed to fulfill exactly the need emerged.

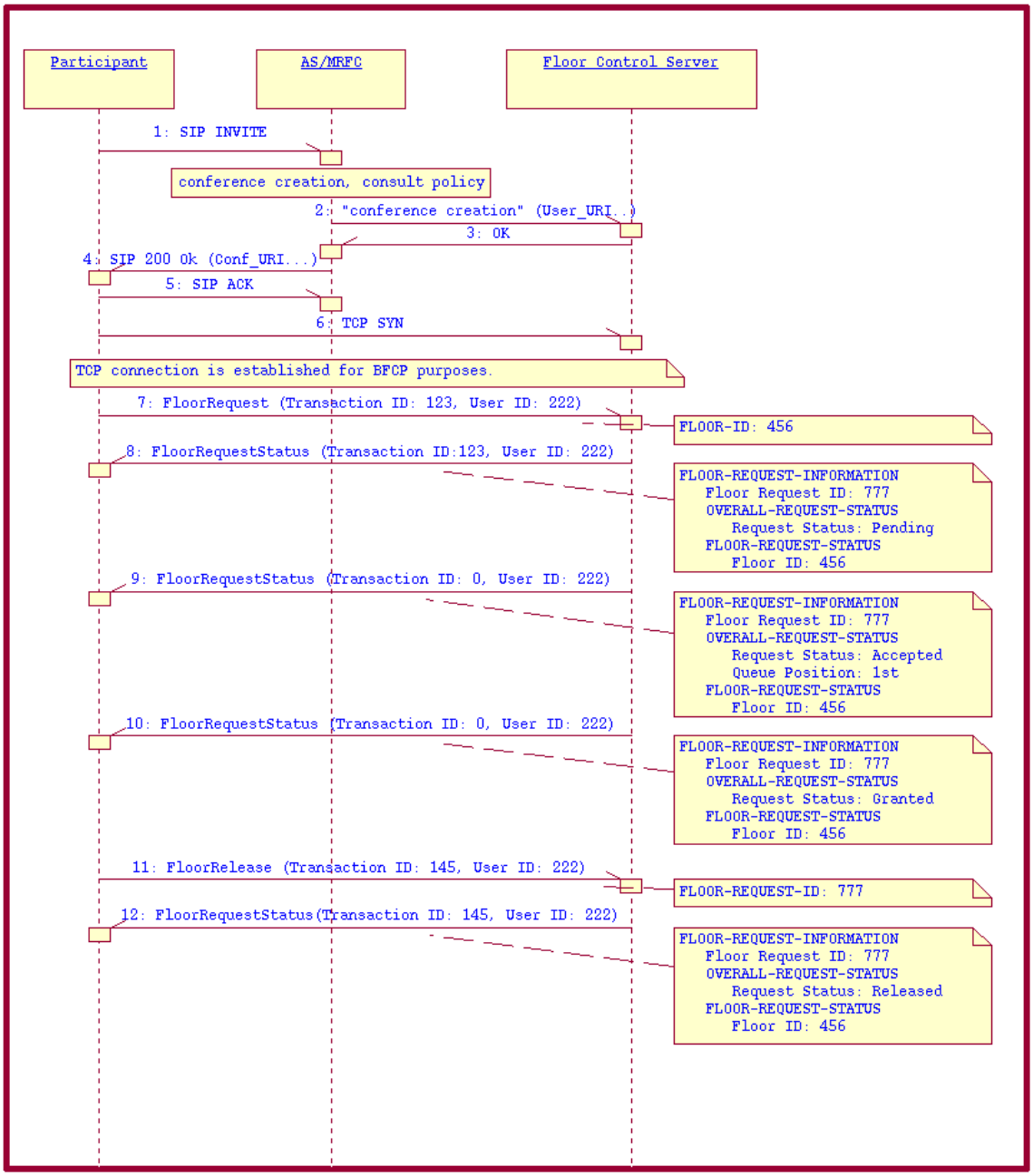

Figure 4-4 Conference set-up and a floor request with a release

5 Control and media plane separation using H.248

The section introduces gateway control protocol H.248 version 3 [Itu05] and the basic mechanisms needed in understanding the Mp interface. The H.248 protocol definitions structure and general aims are explained first, and then the data structures and connection model are described along with the protocol operation.

5.1 H.248 overview

The protocol, also known as the Megaco protocol, is very efficient and it does not require extra overload or heavy syntax. It has two formats available – binary Abstract Syntax Notation One (ASN.1) based and text ABNF based. The H.248 protocol specification brings along two essential abstractions on node level: *the media gateway controller* and *the media gateway:*

Media gateway — The *media gateway* (MG or MGw) converts media provided in one type of network to the format required in another type of network. For example, a MG could terminate bearer channels from a switched circuit network (e.g. DS0s) and media streams from a packet network (e.g. RTP streams in an IP network). This gateway may be capable of processing audio, video and T.120 alone or in any combination, and will be capable of full duplex media translations. The MG may also play audio/video messages and perform other IVR functions, or may perform media conferencing [Itu05].

Media gateway controller — The *media gateway controller* (MGC) controls the parts of the call state that pertain to connection control for media channels in a MG [Itu05].

According to this characterization the MRFC and MRFP share partially this definition, with the wider scope of *call*: communication and logical connectionless or connection oriented association between several peers, generally set up for the purposes of a multimedia conversation. The Mp interface within the media server uses the H.248 protocol. For generality reasons, the perspective of "MGC-MGw" is pertained in this section.

The H.248 protocol is jointly developed with Telecommunication Standardization Sector of International Telecommunication Union (ITU-T) and IETF and it is the standard for allowing a *media gateway controller* to control *media gateways*. Prior to this effort, there were a number of competing protocols, including Media Gateway Control Protocol (MGCP), see Figure 5-1 for the historical dates. H.248 is considered

complementary to H.323 and SIP, in a sense that a media gateway controller will control media gateways using H.248, but the media geteway controllers communicate between one another via H.323[16] or SIP [Pac08].

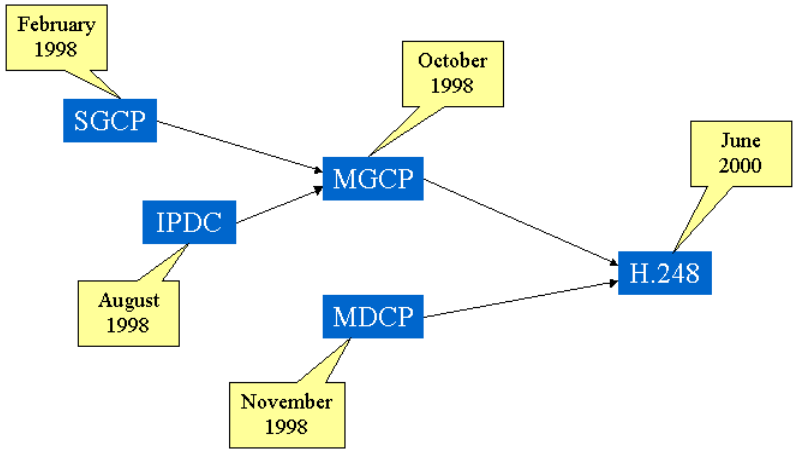

Figure 5-1 The historical dates and protocols in evolution of H.248 [Pac08]

The standardization in ITU-T is currently processing the H.248 version 3 [Itu05], which is the present complete version of the protocol. Version 3 is the one studied in this thesis. The H.248 protocol description references the RFC 2327[17], which is the predecessor of the latest SDP protocol specification RFC 4566 [HJP06]. The latest RFC for the SDP is used in the thesis since the changes between these versions are backward-compatible clarifications or added in light of use. The version change should not jeopardize the protocol usage for floor control purposes.

H.248 is aimed for support of distributing the call control, bearer control and transport into separate entities - signaling nodes in *control plane* to control nodes in the *media plane*. In Figure 5-2 the H.248 usage in IP Multimedia Subsystems (IMS) is shown in the previously mentioned context, marked with red dotted line, in four different situations. The Ericsson IMS Multimedia Telephony Application Server (MTAS) [Eri07b] in picture is a combined application server and MRFC (AS/MRFC) and it communicates with Call Server Control Function (CSCF) using SIP on top of UDP and TCP via the IMS Service Control (ISC) [3GPP07d] interface.

[16] H.323 is another ITU standard for multimedia communication over packet-switched networks. It is an "umbrella" specification, which includes the standards H.323, H.225.0, H.245, the H.450-series documents, and the H.460-series. The scope of H.323 covers real-time voice, video, and data communication.

[17] This is done when the *descriptors* are handled. The descriptors are discussed later in the thesis.

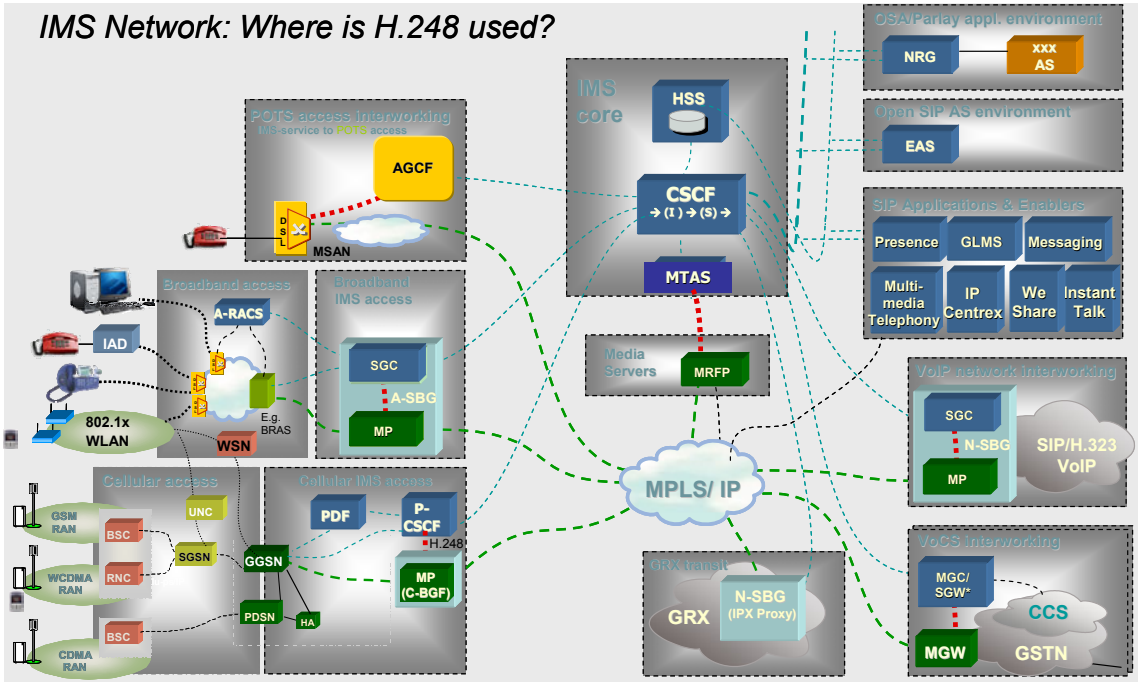

Figure 5-2 Usage of H.248 in IP Multimedia Network.

In circuit switched core network the H.248 protocol is deployed, e.g., in Mc interface between Mobile-services Switching Centre server (MSC-S) and CS-MGw, in Mn interface between IMS-MGw and Media Gateway Controller (MGC), while in IP Multimedia networks it is used in the Mp interface between MRFC and MRFP. A master's thesis "Analyzing the Media Control interfaces and Mobile Media Gateway for IP Multimedia Subsystems (IMS)" [Ram08] around Mp interface gives information on protocol alternatives for the media server and H.248 performance characteristics.

H.248 protocol [Itu05] may be transmitted over IP/UDP using Application Lever Framing (ALF) procedures [Itu05, Annex D.1] or over TCP [Itu05, Annex D.2].

5.2 Protocol definitions structure

The H.248 protocol specification consists of *recommendations*. The main framework is recommendation H.248.1 [Itu05] while the continuously growing number of associated recommendations, now H.248.2 through H.248.59, describe different extension *packages*. Examples of packages are H.248.51 "Gateway control protocol: Termination connection model package" and H.248.4 "Gateway control protocol: Transport over Stream Control Transmission Protocol (SCTP)".

A H.248 *profile* for a media gateway defines how the protocol is used and options for a particular application, e.g., what transport is used and what functionality is supported.

This description with predetermined collection of realized packages, commands and procedures describes the H.248 interface. It is reflection of media gateway *terminations,* which are images of connection points in the ingress, egress and within the media gateway for interconnection of dedicated media streams. Terminations handle the media streams according to very dissimilar characteristics implemented by optional events, signals, statistics and parameters called properties. Terminations diverge by realizing different packages, which the media gateway controller can track down as a whole package or property by property including their current values by using auditing procedures towards the specific termination. A media gateway or a media gateway controller can support multiple profiles and the usage in the reference point between them has to be negotiated.

5.3 Connection model

Termination. The basic building block of H.248 construction is a *termination*. The termination can handle multiple media or control streams[18]. Media stream parameters, as well as bearer parameters characterize a termination. The termination implements signals (e.g. tones or announcements), monitors for events and accumulates statistics. A termination can be of three kinds: *physical*, *ephemeral* and *root* terminations. A *physical* termination, e.g. a Time Division Multiplex (TDM) channel, has a semi-permanent nature meaning that the termination might exist as long as it is provisioned in the node. An *ephemeral* termination is a non-permanent channel, which existence is limited to the time of its use. An ephemeral termination is created and destroyed with a command. The *root* termination has a special meaning – the gateway or node itself can be referred as an entity by using this termination type. The unique termination identity called "Root" may have also properties, events, signals, statistics and even protocol packages defined for it.

Context. A *context* is a relationship of one or more terminations and represents internal bearer connectivity. The context is an association between terminations in a collection of H.248 terminations[19]. The context has topology attribute giving the flow directions between terminations of context, namely *isolate*, *oneway* and *bothway* as defaulting value, specifying who hears or sees who. A termination can belong only to one context at a time. The typical context can have only two terminations, but some media gateways implement multi-point connectivity e.g. for conferencing or handover purposes, which allows more terminations per context. See Figure 5-3 as an example for such a context with topology definitions. In the picture the *user 1* and *user 2* are in a conversation

18 Here the stream is a *bi-directional media or control flow received/sent by a media gateway* (or endpoint) *as part of a call or conference* [Itu05].

19 A special type of context, the null context, contains all terminations that are not associated to any other termination. E.g., idle subscriber lines in a gateway can be represented as terminations in the null context.

relationship with each other. The *user 3* is listening the conversation from the termination (*T1*) representing the connection point of *user 1* towards the call without possibility to be heard by the other call participants. It looks very much like the *user 3* would be eavesdropping the call since there need not be any information available for the *user 1* and *user 2* about the situation.

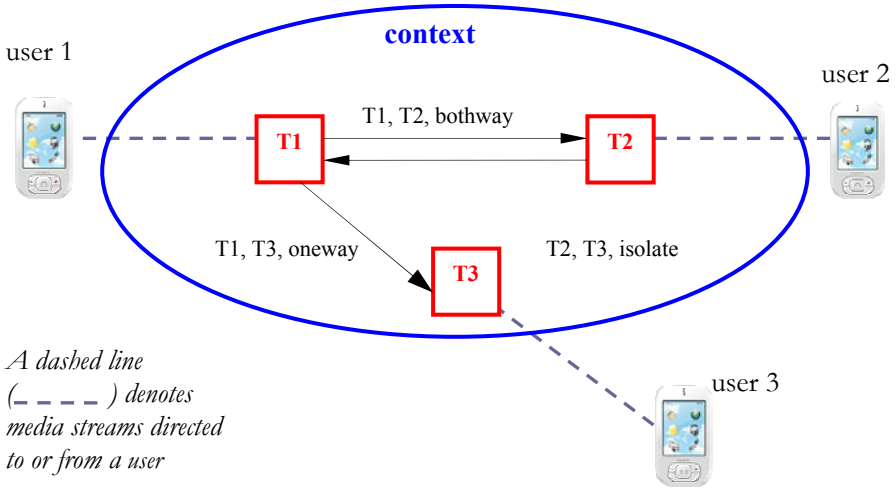

Figure 5-3 A context with three terminations, context internal limitation of the streams

The *Stream Mode* stream property (visible in LocalControl Descriptor in Figure 5-4) of a termination determines the external behavior of a context in relation to a specific termination. The mode affects only media, not signals or events. The mode alternatives are *send-only*, *receive-only*, *send/receive* and *inactive*, which each mandate the allowed direction or directions for the media flow whilst inactive forbids all media traffic for the termination.

5.4 Data structures

The message syntax may follow binary ASN.1 encoding [Itu07, Annex A] or textual format [Itu07, Annex A] allowing both "pretty" and "compact" token formats. The data structures specific for H.248 include specific message construction. Command

parameters are structured into *descriptors*[20], some of which are distinctive for specific commands. The general text format of a descriptor is

```
DescriptorName=<descriptorID>{parameter=value,  parameter=value…}
```

In core H.248 when using text encoding for the protocol, the media descriptors' local and remote descriptors consist of session descriptions as defined for SDP[21]. The binary format uses the values for parameters from SDP specification. The main descriptor, the media descriptor is constructed as depicted in Figure 5-4. The table of all descriptors with short explanations is found in thesis appendix [A.6] and full descriptions of those can be found in the H.248 specification [Itu05].

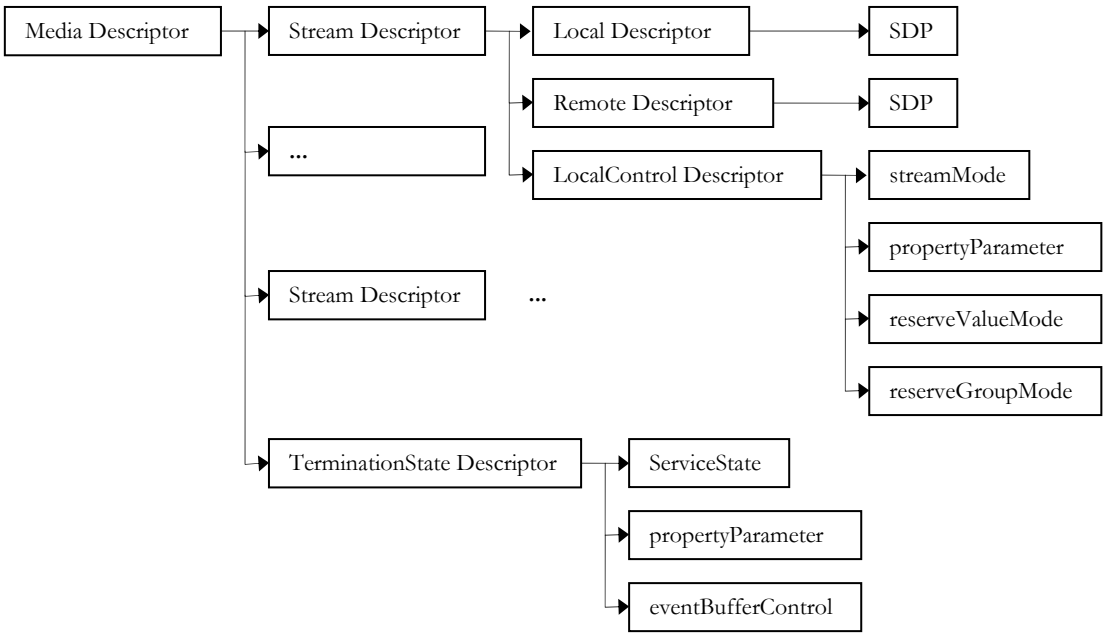

Figure 5-4 H.248 media descriptor construction

A parameter or complete local or remote descriptor may be considered in H.248 commands as *fully specified*, *overspecified* or *underspecified*. *Fully specified* parameters have an explicit value in the command that the command responder from now on is instructed to use. *Underspecified* parameters, indicated by the "CHOOSE" value, let the command responder to choose the parameter value. *Overspecified* parameters have a list of allowed values which reflects the command initiator's order of preference. The command responder chooses a value from the list and indicates it to the initiator.

---

20 This is not to be confused with SDP media descript*ion*, which is a part of the SDP session description.
21 SDP is used in H.248 with small deviations [Itu05] not visible in the level of this protocol outline.

The *TerminationState* descriptor contains common *termination* properties, including *ServiceState* property which tells whether the termination is in or out of service. The termination properties are recognized by unique IDs. Their default values for new terminations or terminations in *null context* are set in the standard, set by provisioning, set in the extension package if the property is not defined in the standard. In case none of the former applies, the values default to "no value" or empty. *Media streams* have properties for both directions separately in local and remote descriptors[22] for media gateway incoming and outgoing streams. The local control descriptor lists the locally supported properties. The properties can be read-only or read-write from media gateway controller point of view and they can also be defined as global for an extension package.

5.5 Operation

Call control is fully handled by the media control plane. The H.248 connection model introduces two main abstractions in the user plane not dependent on call state – the termination and the context. The media gateway controller which is aware of the call state can control the terminations and the contexts in the media gateway using commands.

The H.248 message identifies the sender and the protocol version, but basically the rest of it is merely concatenation of transactions. The message decomposition is given in Figure 5-5; an example of a text format message is given in appendix [A.7] of the thesis.

A transaction, treated independently within a message, consists of collection of commands to which the replies are to be delivered together. The commands are handled in order of reception within a transaction and are further grouped into actions, each action concerning only one context. The H.248 transaction application programming interface (api) describes three transactions: *TransactionRequest*, *TransactionReply* and *TransactionPending*. A request transports actions between nodes, a reply transports the command responses and *TransactionPending* indicates that the execution is already ongoing and no request retransmissions are needed.

[22] non-core protocol support is expressed as packages

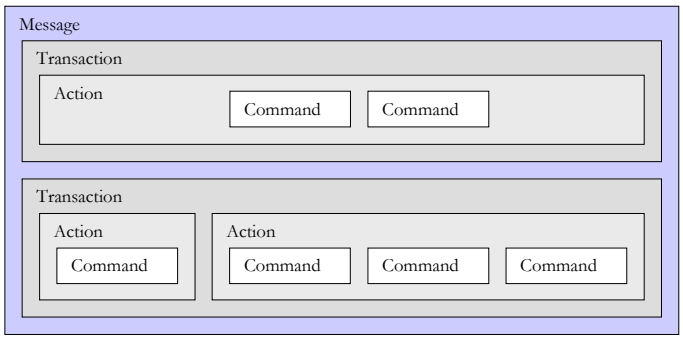

Figure 5-5 H.248 grouping of commands

The initiator of the message is in most cases the media gateway controller. The media gateway uses the command *Notify* to inform media gateway controller, while the *ServiceChange* can be sent by either node. Below is an overview of the commands [KKS1].

Add The *Add* command adds a termination to a context. The Add command on the first termination in a context is used to create a context.

Modify The *Modify* command modifies the properties, events, and signals of a termination.

Subtract The *Subtract* command disconnects a termination from its context and returns statistics on the termination's participation in the context. The Subtract command on the last termination in a context deletes the context.

Move The *Move* command automatically moves a termination to another context.

AuditValue The *AuditValue* command returns the current state of the properties, events, signals, and statistics of terminations.

AuditCapabilities The *AuditCapabilities* command returns all the possible values termination properties, events and signals allowed by the media gateway.

Notify The *Notify* command enables the media gateway to inform the media gateway controller of the occurrence of events in the media gateway.

ServiceChange The *ServiceChange* command enables the media gateway to notify the media gateway controller (MGC) that a termination or a group of terminations is about to be taken out of service or has just been returned to service. *ServiceChange* is also used by the media gateway to announce its availability to an MGC (registration), and to notify the MGC of an impending or completed restart of the media gateway. The MGC can also announce a handover to the media

gateway by sending it a *ServiceChange* command. The MGC also uses *ServiceChange* to instruct the media gateway to take a termination or group of terminations in or out of service.

The commands provide control over the properties of terminations and contexts, including the context topology, event reporting and the signals and actions towards a termination. The commands *Add, Modify* and *Move* can change the values of the termination properties. The values of the wanted changeable parameters are given in respective descriptors when modifying, moving the termination or adding the termination into context, while the parameters not mentioned keep the former values. The descriptors can be returned in some occasions as a result of the command.

An example of a call setup and release from H.248 point of view is depicted in Figure 5-6. The media gateway controller (MGC) receives a call initiation message from network. It reserves a context (CTX1) and terminations for incoming (T1, A) and outgoing (T2, B) connections from the media gateway (MGw) using *Add* commands, to which the media gateway answers with replies (as to all commands in this flow). At this point the calling subscriber A is to be played a ringing tone and he can send and receive streams while the called subscriber B can not hear nor send anything. The called subscriber gets indication about an incoming call from his end of the network and answers. That is indicated to the media gateway controller with "B-answer" which makes it to request to stop the ringing tone to A and to set the B to listening mode with *Modify* commands. When the media gateway controller gets indication from throughconnection from the network the call is throughconnected from A to B with a *Modify* command by manipulating the stream mode towards the called subscriber to allow him also to send streams. Now the media can flow to both directions. After a while, the caller terminates his side of the call by hanging up which is signaled to the media gateway controller. It orders the media gateway to release the resources connected to the incoming and outgoing terminations with *Subtract* commands. After the terminations are removed, the media gateway automatically deletes the empty context and the call is permanently cleared.

The message flow explained is taken from a specific working call setup case for testing environment; therefore the messages consist of only one command per each message. The scenario is valid, but can be compressed by using less messages where appropriate, e.g. for the *Add* requests when reserving terminations and the initial context.

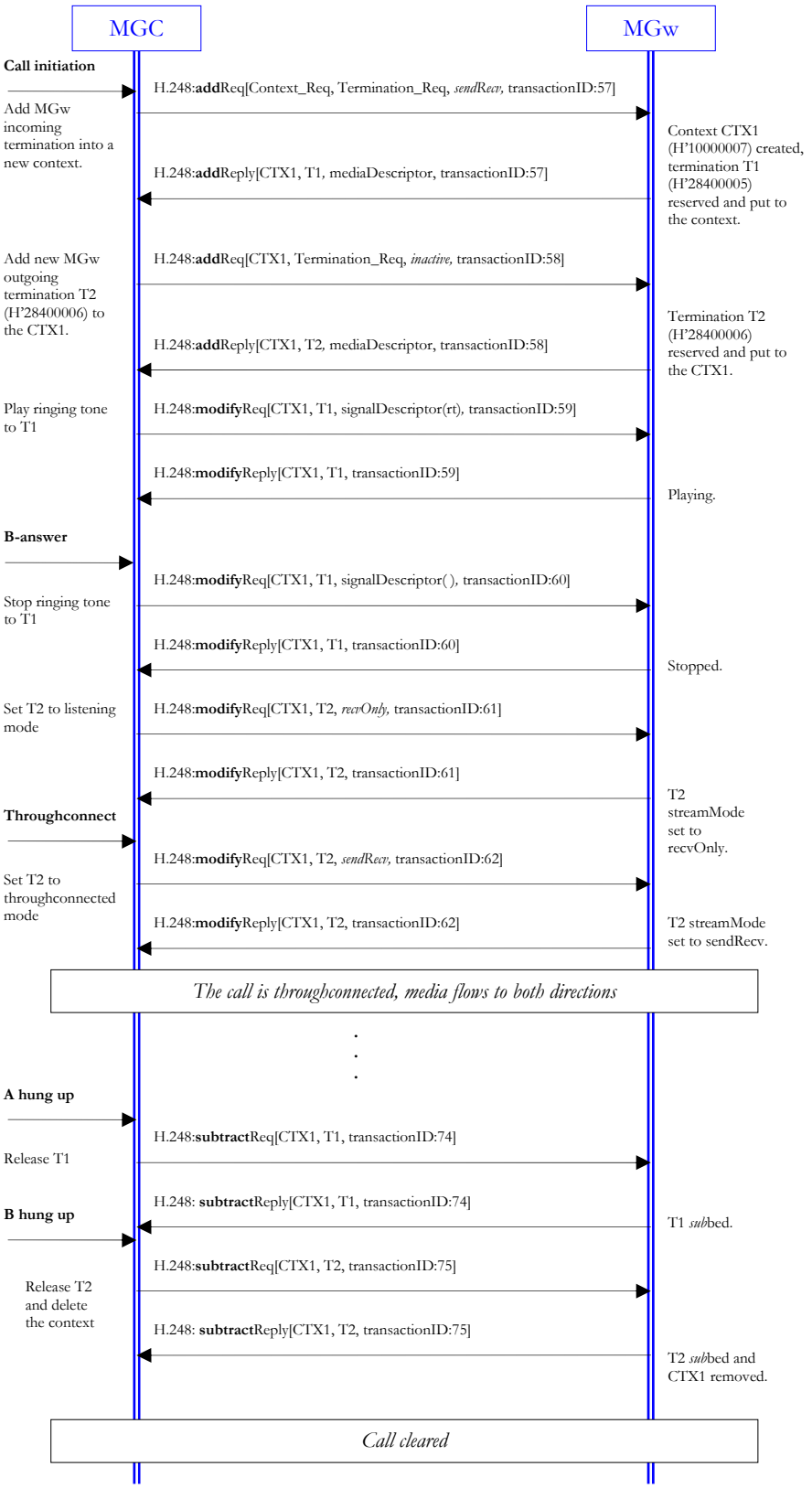

Figure 5-6 H.248 call setup and release.

6 The impact of floor control on Mp interface

This section explores the Mp interface between the Multimedia Resource Function Processor (MRFP) and the Multimedia Resource Function Controller (MRFC) focusing on different locations for the floor control server (FCS). The H.248 protocol is studied here in respect to the interface primitives and parameters concerning normal conference scenarios. The support for multiple MRFPs separated for different multimedia types is evaluated [23]. First we examine the present situation on standardization and set assumptions and base for the analysis. Then we depict the current specification approach from 3GPP to set the floor control server in the MRFP side in the media server; the requirements for floor control parameters regarding service activation and usage are studied. The offered packages for the service are looked at in contrast to the requirements analysis. Then we give a flow for a simple use case introducing changes, accompanied with explanations, alternatives and a summary. In addition we investigate the two cases where multiple streams would be handled by different MRFPs, with perspective set by cascaded conferencing models [Nov06]. In the last subsection we examine a scenario, where the floor control server would be located not in MRFP but in MRFC. The same aspects are used in evaluation and requirements analysis for both of the network scenarios.

6.1 General

The current 3GPP specifications for conferencing give no descriptions how tightly-coupled conferencing modifies the Mp interface [3GPP07g]. To be specific, the technical specification for conferencing using IP Multimedia [3GPP07c] states that the H.248 signaling between MRFC and MRFP is not covered, and the separate specification 23.333 for the Mp interface [3GPP07f] includes only the unscheduled on-the-fly formed ad-hoc conferencing in current 3GPP release. The 3GPP is making a change request [3GPP08] to the Mp interface procedures descriptions at the moment to cover floor control and support for other kinds of conferencing is scheduled to be incorporated in the upcoming release 8. Nevertheless, as the change request for 3GPP specification 23.333 [3GPP07f] is the most complete information from 3GPP, that is used as input where feasible together with the approved 3GPP specifications for rest of the area under discussion, completed with the up to date IETF work. Furthermore, the definitions of cascaded tightly coupled conferences are reused when the possibility of multiple MRFPs separated for different medias is studied. The subject of cascaded conferences itself is investigated in "Study of Cascaded Conferences and

[23] The possibility that a Multimedia Resource Broker (MRB) [3GPP07e, pp. 29-31] between AS and MRFC would be deployed in the network is not considered nor discussed in the thesis.

Implementation of the Binary Floor Control Protocol (BFCP) for Centralized Conferences" [Nov06].

The H.248 extension package H.248.19 [Itu04] gives aids for audio, video and data conferencing for decomposed multipoint control unit (MCU)[24], but since the floor control part was introduced before the definition of floor control itself was fixed, the package does not cover the needed means and it is partly misleading in its inadequacy. The Amendment 1 for the H.248.19 exists from year 2006, but it does not give any help to the situation. The latest Amendment 2 work item [Itu08] comprises the specification work in ITU-T going on in this area to add in the floor control. The paper is in draft phase and the results, aligned with 3GPP and IETF work concerning floor control, might not be available in the end of the year 2008 or in 2009.

Figure 6-1 gives a picture on fresh 3GPP view on floor control and its role-players' positioning in relation to media server, denoted as the Media Resource Function (MRF) consisting of the MRFP and MRFC. For comparison, see media server decomposition in Mp interface specification [3GPP07f] and in Figure 3-2. Note, that even if the actual place of the floor control server is only hinted in this picture to be in MRFP, the approved 3GPP specifications fix this location seemingly without questioning.

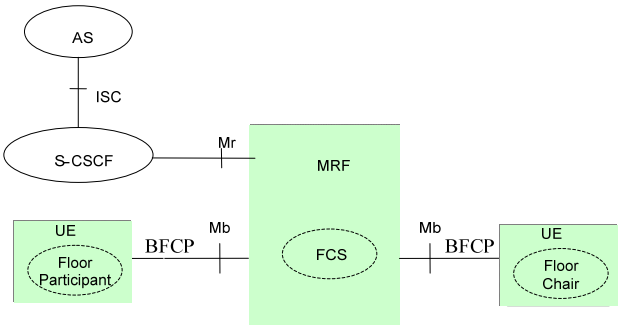

Figure 6-1 Functionality architecture of floor control [3GPP08]

There are three possible options when considering the location of the floor control server and BFCP termination within the media server. When the floor control server is situated in the combined MRFC and conference application server (AS/MRFC)[25], the TCP/BFCP connection towards the conference participant can be in the MRFP with

[24] The decomposed MCU is constructed of media controller and media processor dedicated for conferencing and the complex is logically mapped to media server (MS) by the 3GPP, the parts equivalent to MRF Controller and Processor, respectively.

[25] The picture shows more nodes in between but logically the AS/MRFC comprises the controller part of the Multimedia Resource Function (MRF) and the SIP conference application server (AS) where the existence of serving call server control function (S-CSCF) is not relevant for floor control functionality.

BFCP relayed in the Mp interface or instead the TCP/BFCP connection can be directly in the AS/MRFC. When the floor control server is collocated with MRFP, it is most natural to have the TCP/BFCP connection there, close to the floor control server.

When taking into account TCP connection management load, load sharing principles of MRF, and limited extensibility for BFCP transport, the TCP/BFCP connection in AS/MRFC is not considered a real alternative for debate. Still, if the combination of both floor control server and BFCP termination in AS/MRFC would be taken into account, it would in general mean more traffic through the Mp reference point in certain cases. The media modifications and requests regarding tone[26] sending would stress the Mp interface most when comparing to non-floor-controlled conferences. Higher rate in media manipulations would be apparent in cases where floor control affects the participant's stream mode. Special indications induced by floor control provided for the user from MRFP and not by the user equipment (UE) as reactions to received BFCP messages would increase tone sending requests. This rules out the choice of having both the TCP connection and the floor control server in the MRFC.

We are now left with two distinct selections for analysis clearly changing the Mp interface communication, both having the BFCP terminating in MRFP: floor control server in MRFP or floor control server in AS/MRFC. These two choices are discussed in the next subsections.

The chair role. The conference chair is substituted in the models reflected in the message flows in this section with a first-come-first-served (FCFS) queuing algorithm, with visible granularity reminding the first-in-first-out (FIFO) queue functionality. This is done due to the fact that in the choice where the floor control server is in MRFP, the chair actions itself do not add more messages to Mp interface than granting, releasing or revoking a floor do. See Figure 6-2 and Figure 6-3 for reference for BFCP *Chair action procedure* and *Floor releasing and canceling procedure*, where UE#2 is acting as chair. In Figure 6-2 the UE#1 first requests a floor. Since the UE#2 happens to be pointed as the chair of the conference, the floor control server informs the UE#2 about the floor request in a *FloorStatus* message and gets an answer back in a *ChairAction* message. The floor control server acknowledges the message and sends information on chair decision, whatever it is, to the UE#1 in *FloorRequestStatus*. Here the decision seems to be, that the UE#1 gets the floor and he is allowed to send and receive conference streams. In Figure 6-3 the UE#1 releases the floor which is informed to the floor chair. A *FloorRequestStatus* is sent to the UE#2 as confirmation.

---

26 The tones are defined according to the package [Itu08] as: "Depending on the media type, these tone indications may be a tone, an announcement, text, still or moving image which is provisioned on the media processor"

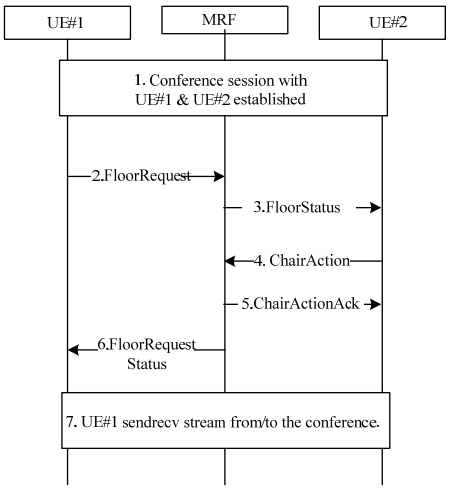

Figure 6-2 User requesting the floor during a conference [3GPP08]

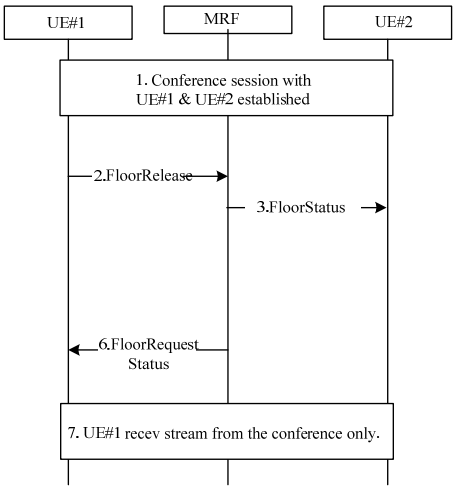

Figure 6-3 User releasing the floor during a conference[3GPP08]

In the choice where the floor control server is in MRFC, the BFCP messages are carried through the Mp interface in H.248 messages when the BFCP connection is also in the MRFP. In any case, the load increase connected to having chair or chairs is dependent on floor requests and since it is subject to great fluctuation, it is decided not to be shown separately in message flows. It is assumed that the floor request queue is implemented in the floor control server rather than in the application server to reduce the message flow between the components.

The approach towards media modification. The BFCP message flows discussed in previous sections do not take into account the possibility, that the floor control decisions could change the stream modes for participant medias, which is a highly recommended alternative according to floor control protocol requirements [Kos06] and it is also driven by the 3GPP. The BFCP specification implies [COD06, p. 8 "Privileges of Floor control"], that the protocol does not mandate which participants actually can exercise their rights to use the conference streams, and refers to the XCON conference framework [BBL07] for information on media manipulation. The framework refers to *1st party signaling*[27] and, as an alternative or addition to it, instructs to implement a conference control protocol client aimed for resource manipulation. The XCON framework introduces also a new participant role, *moderator* [BBL07], who could act on other participants' behalf. The conferencing and floor control related H.248 packages [Itu03, Itu08] give means to indicate the conference and floor information to the participants, which should be able to be utilized also without media manipulation as characterized in BFCP specification [COD06]. Performing media manipulations as an alternative is further in the thesis considered on what comes to BFCP procedures.

[27] The participant can send a SIP re-INVITE request with an altered session description to focus. It does not affect states of other parties in the conference.

On flow boundaries. Not all BFCP entity transaction procedures trigger actions on Mp interface. The procedures handled solely in the FCS are *Floor information procedure, Floor request information procedure, User information procedure* and *Server capability information procedure*. The *Chair action procedure* has impact on the interface, but as mentioned before, it is not taken up as separate subject. The remaining two procedures, *Floor request procedure* and *Floor releasing or floor cancelling procedure* are visible in the upcoming flows. Whether the TLS connection is used with the TCP connection is irrelevant considering the thesis scope and therefore is not shown. The messages are given using only the relevant information and the messages are considered to be only for one specific conference. Conference indications, including tones or icons visible for a participant, e.g., for signifying floor holder or a queuing floor request, are provided by the user equipment based on BFCP signaling. The conference entry and exit tones are provided for the participant by the MRFP.

6.2 Floor control server in Multimedia Resource Function Processor

Requirements analysis. A firm basis for analysis is the floor control protocols requirements specification [Kos06], parts of which are visible in the Mp interface. Some emerged requirements are also considered for consistency. For a number of names and values there are acronyms introduced for easier reference. Additional signals and tones are not considered here, since the available conference enter and exit tones can be used and all other indications can be produced by the user equipment (UE) as reactions to BFCP messages.

Let us first examine, what parameters are needed in H.248 protocol for basic service activation, not assuming any H.248 packaging. Firstly, there must be a parameter introduced as H.248 property that indicates whether the floor control is in use or not [Kos06, REQ-F1]. At the same time it could be pointed out which MRFP has the floor control server for the conference. An enumerated property "Floor Control" (*fc*) in the TerminationStateDescriptor can be used to get the desired effect. First value of it indicates that the termination is not part of a floor-controlled conference (*nofc*); second value "active floor control server" (*afcs*) indicates that this MRFP has as the floor control server for this termination. The third value "floor control message tunneling" (*fct*) indicates that the floor control server for the termination is in the MRFC and all the BFCP messages should be relayed through the Mp interface. The fourth value "remote floor control server" (*rfcs*) indicates that the floor control server is in another MRFP. There are two aspects in setting the property - to set the floor control server per conference or as a fixed situation, probably depending on what is the outcome from standardization - the property could be set for the user termination or *root* termination. In addition, the streams for user should have indicators telling if the mixing for the stream is to be done locally to determine the stream "owner" and if the

media stream mode is to be exposed to floor control decisions. These two last indicators can be put on termination or *root* level, like the *fc* property.

Next requirement is to identify the algorithm for granting the floor [Kos06, REQ-F2] which could be an enumeration, "Floor Control Algorithm" (*fca*), of "chair controlled" (*cc*), "FCFS" (*fcfs*) indicating *first-come-first-served* FIFO-like algorithm or "random" (*rand*), whereas rest of the enumeration is to be filled with appropriate algorithms as needed. In conjunction with this, there should be properties for indicating a chair and possibly a moderator [Kos06, REQ-F6, REQ-F7]. Whether the chair and the moderator are the same role or not is an open question – a termination could have a boolean property "Is Chair" or "Is Moderator" to indicate whether the termination in question is chair or moderator for that stream. In case the moderator is a separate role, a property "moderated" (*mod*) must also be included for a stream. The requirement for setting the boundaries for simultaneous floor holders [Kos06, REQ-F3] adds another parameter, "concurrent floor holder limit", for the stream. The properties for granting algorithm and determining the specifics of chair and moderator should be added into the LocalControlDescriptor.

An event, telling the event requestor (MRFC) that the floor status has changed, must be introduced, e.g., in case charging, stream modification or updating the conference data is needed. The ObservedEventsDescriptor for a termination must comprise the floor statuses *granted*, *released* and *revoked* associated with the streams for the floor.

Support from standardization drafts. The standardization results this far offers the draft version of H.248.19 Amendment 2 [Itu08] which gives some support for floor control server realized in MRFP. The event is introduced in a way that matches the above analysis. The requirements for floor granting algorithm [Kos06, REQ-F2] and "concurrent floor holder limit" [Kos06, REQ-F3] are also harmonized, rest of the requirements handled above are introduced in a slightly different way. The amendment also brings in features not discussed here, e.g. for limiting the floor keeping time and some additions in area of tones. The packages in the amendment for optional floor control support in the MRF are "Floor control" for generic support, "Floor Action" for floor information and conference status indications, "Indication of being viewed" for visibility information, "Floor Control Policy" and "Floor Status Detection". When the floor control is chosen to be used according to these packages, at minimum the package "Floor Control" has to be supported in media server.

Scenario description according to requirements analysis. In this use case we assume that the floor control is taken into use in conference level as opposed to node level. To indicate that the floor control server is active in the MRFP, the MRFC commands the MRFP to indicate "active floor control server" in the property "Floor

Control" in the TerminationStateDescriptor for each new termination in the conference, along with appropriate data including the suggested algorithm for granting the floors. In case the different BFCP status messages do not trigger any indications at user equipment (UE) on conference stages, to get indications to participants on changed conference and floor status the MRFP has to notify the MRFC. The indications provided for a user could be an announcement or a warning tone etc. when the conference roles change with e.g. granting of a floor. The requesting procedure is done with the new event detection requested for all the conference terminations, because of which the MRFP informs the MRFC with a detected event in a *Notify* procedure. The MRFC has to be notified also in case floor control decisions modify the stream modes of the participants involved. The FCS using MRFP has to request MRFC, similarly with a *Notify* procedure, to update the modes. In both modifying cases the H.248 *Modify* request procedure is concluded for the terminations involved, and the SIP/SDP offer/answer exchange is not needed, but it can be performed prior the modifying. The conference object may include optional information on floor and floor holders, and as the object is situated with the conference focus, updating via Mp interface is needed. Updating the conference object can be done basing on information on the *Notify* request already supplied, providing an indication to participant or stream modification is also wanted. Otherwise, a dedicated *Notify* request is needed.

There is an alternative to the aforementioned procedure under discussion, where an MRFP could make decisions autonomously (without an MRFC) on changing stream modes and maybe even offering indications, based on floor control information. The course of action would need only a Notify procedure towards MRFC from MRFP with above mentioned information. The alternative with this in some extent autonomous MRFP might though violate the H.248 *master-slave* –principle - even if SDP offer/answer exchange [RoS02] would be used in addition.

To wrap up, there must be a Notify request from MRFP to MRFC during a conference when streams are to be modified, tones are to be sent to the user or conference data is to be updated in concert by both media server functionalities. Only the conference data update does not trigger modification procedure from the MRFC when receiving the Notify, but it is the only one of the three abovementioned actions that has to be done when the criteria are met even if the MRFP was allowed to act autonomously concerning sending indications to users and modifying streams. Note, that the conference indications in user equipment can be triggered by BFCP messages alone.

Conference set-up and release, a simple case. Let us first consider the message flow in Figure 4-4 from Mp interface perspective. It is not a typical full scenario since it depicts a conference involving only one user, nevertheless, it gives a base on the Mp interface evaluation and can be looked as subject to modification to show the needed

messages and a probable flow. To be more realistic, the message flow in Figure 6-4 is complemented to have two participants; subsequent participants behave as the 2nd participant. For simplicity, the users are expected to invite themselves as *dial-in*[28] users that learned about the conference from external source and the conference data is assumed such that it needs no modification from BFCP procedures. For clarity, modification of stream mode is not considered. Thus floor requests do not involve messages on Mp interface and the terminations are set from the beginning to stream mode *sendRecv*.

In Figure 6-4 the AS/MRFC is first triggered with SIP invitation to create a conference and the first participant dialog using a remote address, conference URI and BFCP set-up parameters within a session description (SDP). The SDP mediated must have an attribute *a=floorid* that not only indicates the requested floor identity but also gives the stream or streams associated with labeling [LeC06] to that specific floor, here the single stream, indicated as to be mixed locally, is named shortly as "strm 5" to preserve generality. The TCP connection is announced to be established from the participant side using *a=setup:active* in the remote SDP. The MRFC issues an *Add* command towards the MRFP using freshly supplied information in order to create the conference context and the first termination in it. The termination *a* and context identities are given as reply, and the local address, media descriptions for the TCP termination with other connection data are sent towards the participant according to SIP/SDP offer/answer model. When a user becomes a conference participant, a tone[29] indication *Conference entrance tone* (*enter*) according to "Conferencing tones generation package" [Itu03] and "Floor Action" package [Itu08] may be requested with *Modify* to be supplied for the participant. In case there is an additional package defined, which includes conference warning tone, it could be requested as well to last up to conference exiting, in this case when the participant voluntarily leaves the conference the UE handles the tone. The participant initiates the TCP connection with floor control server and starts using BFCP interface by first requesting the floor. After the floor is granted for *A*, another user decides to join the conference. The termination *b* is created and handled in same manner for the participant *B* as for *A,* but it is added to the existing conference context. Also the participant *B* asks for floor and is inserted into queue to wait for floor release from *A*. Whenever *A* gives up the floor, *B* receives a situation update and gains the floor.

The conference continues like this; participants can join and leave the conference until the last participant leaves, when the conference - its resources and data structures - are

28 When the new participant is inviting himself, he *dials in* to the conference, the full remote SDP is given in the invitation. *Dial-out* user terminations are added before the SIP 200 Ok response arrives with the full remote SDP, therefore modifying the termination is needed when the response is received.
29 A tone is defined in the package [Itu08]: "Depending on the media type, these tone indications may be a tone, an announcement, text, still or moving image which is provisioned on the media processor"

released. A user release from conference system in this flow is done from participant's initiative within an existing SIP dialog[30] with a standard *BYE* transaction. If the release of a participant would be initiated from conference system, a *Conference exit tone* (*exit*) according to "Conferencing tones generation package" [Itu03] and "Floor Action" package [Itu08] would be requested with *Modify* to be played until the SIP dialog had concluded.

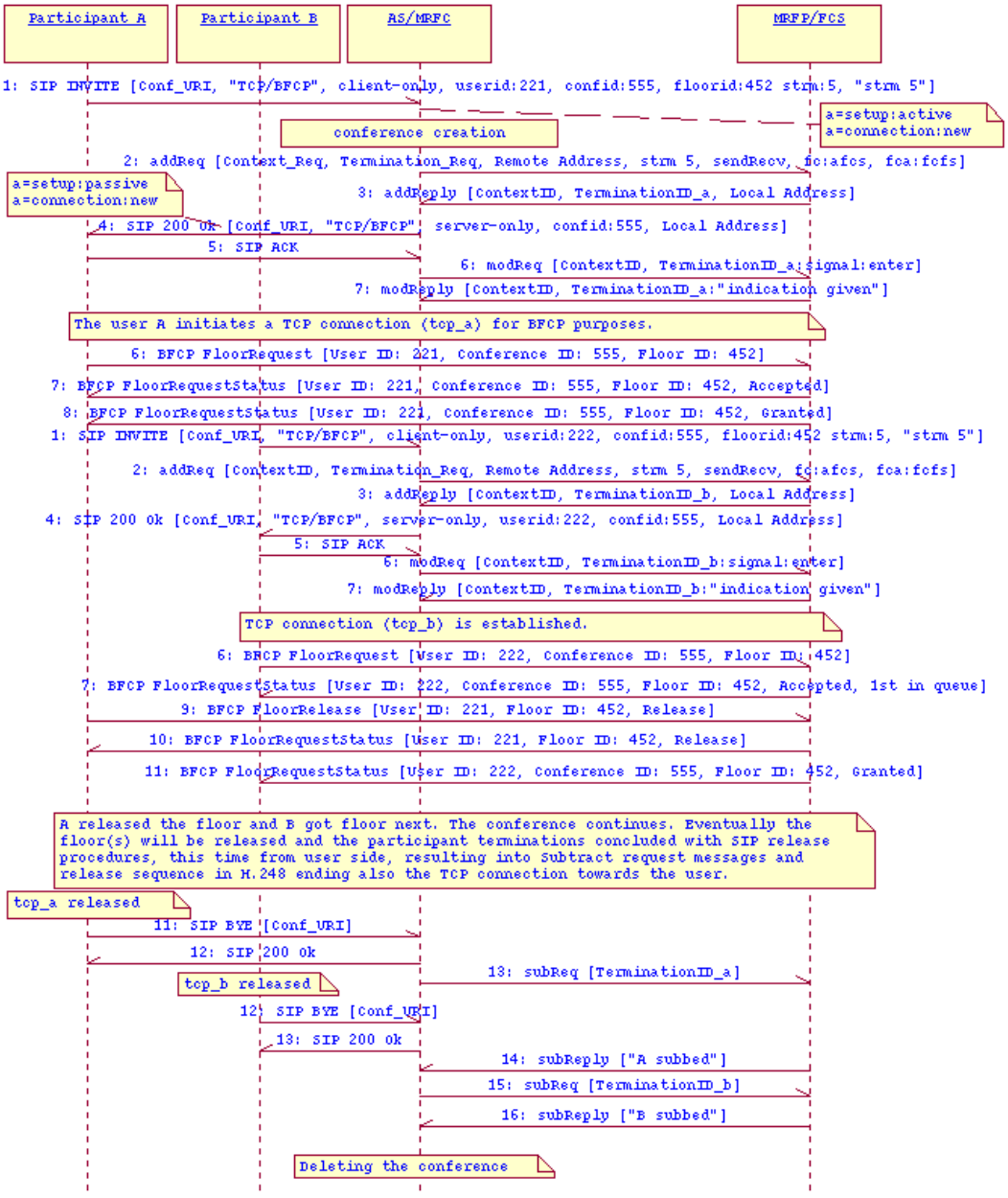

Figure 6-4 FCS in MRFP: Conference set-up and release in Mp interface

[30] The *To*-field of the *BYE* is tagged to identify the dialog.

Alternatives and flow summary. Now in the introduced successful scenario the Mp interface is used for each participant attending the conference in following order:

- *Add* request for creating a termination and a context (only for the first) and a reply to it. The SDPs received from the user are applied in Remote Descriptors for the streams requested,

- *Modify* request to get conference enter tone for a termination and a reply to it. When the user is a *dial-out* user[31] (the conference *focus* invited the user), the final configuration and media resources received in SDPs of a SIP *200Ok* from the user can be sent also in this *Modify*,

- *Modify* request to get conference exit tone for a termination and a reply to it if the participant removal is initiated from the conference system side,

- *Subtract* request to remove the termination when a participant leaves the conference and a reply to it.

Including the stream modification for floor holder changes would in most cases multiply the Mp interface load, when looking at the case where the MRFP has to get orders from the MRFC to change stream modes. The addition would be a single *Notify*[32] towards the MRFC and the pair of a *Modify* and a reply for every granted floor request in addition to abovementioned messages[33]. The minimum message count rises from six to nine with addition of six for every subsequent attendee according to the participation count for the conference. The minimum message count represents the situation, where one participant is giving a lecture without releasing the acquired floors and hence no other requests for floor. Maximum message count has no theoretical upper limit – active floor requesters and releasers in numerous concurrent conferences within the same FCS can keep the Mp interface rather busy. In case an MRFC must be separately requested to command various incidents, addition of the three (*Notify*[34], *Modify* and *Modify* reply) with appropriate information would be needed for each occurrence.

When the conference data includes floor information such, e.g. floor holder, that floor change has to be informed to the AS/MRFC, every floor grant triggers one *Notify*[34] towards the MRFC, the minimum message count changes to seven. Only in case the streams are modified also, the floor change does not trigger any extra messages as it can

---

[31] The conference system invited the user

[32] Single or accompanied by a reply, depending on implementation.

[33] The initial value of stream mode for a termination would logically be *inactive* which would be set to *sendRecv* from the first received floor

[34] Single or accompanied by a reply, depending on implementation

be piggybacked in one and the same *Notify* and message counts can be kept as for the stream modification case.

Multiple streams handled in different MRFP/FCSs. A problem is seen in a situation, where the resources for one or some of the streams are handled by a remote processor: the stream handling regarding is distributed but the floor queuing for the streams of a conference is defined now as centralized in a local floor control server. A solution to this could be, that the MRFPs for different medias would all have separate floor control servers and the local floor control server could handle the floor requests for the remote resource. This could look like the chair for the stream would be located in that remote processor[35]. The local termination towards the chair from both sides could be created at time, when the AS/MRFC notices that the conference is to be set up such that multiple MRFPs are needed.

The remote floor control server would be treated as in *full mesh cascaded conferencing* or *hierarchical cascaded conferencing* model [Nov06], shown in Figure 6-5 and Figure 6-6. The models handle a media server in complex with an application server (AS/MS) as a whole where the focus and FCS are collocated. The cascaded conferencing models introduced are distinguished on the way they link their focuses, hierarchically or with having all connections to each others.

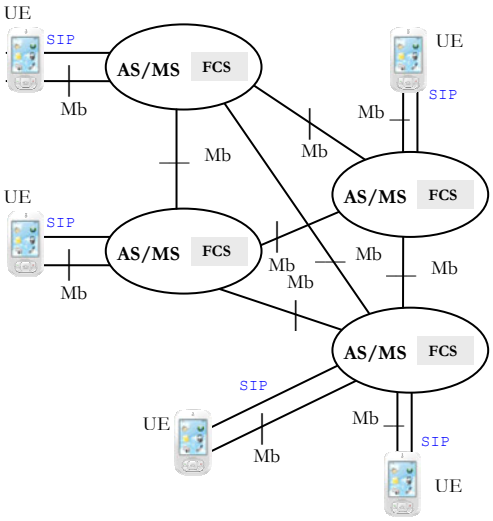

Figure 6-5 Full mesh cascaded conferencing

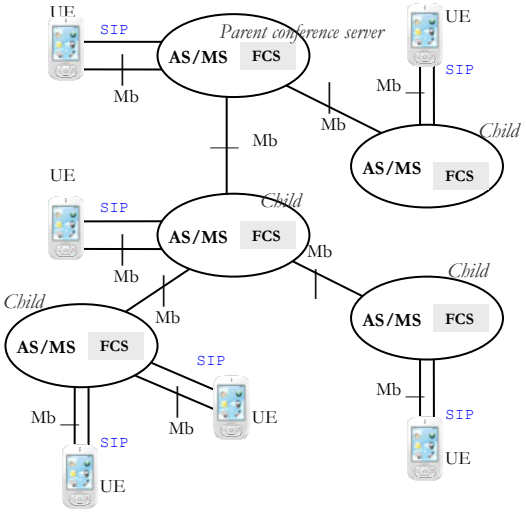

Figure 6-6 Hierarchical cascaded conferencing

[35] The terminations connecting the MRFPs should have in both ends the *"Is Chair"* property set for the appropriate streams, while both ends should indicate *fca:cc* for the matching remote streams for all terminations.

Since we are discussing about floor control, we refer to the MRFP hosting FCS in the distributed media server keeping only *one* AS/MRFC and introducing the construction of the network from that perspective, depicted in Figure 6-7.

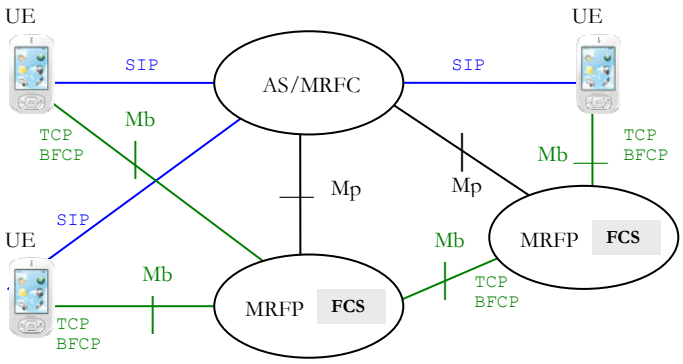

Figure 6-7 Multiple streams handled in different FCSs

The local FCS receiving floor requests for a remotely floor-controlled and owned[36] stream would issue a FloorStatus message towards the remote FCS, where the message would be treated as a request for chair to grant floor and put as a floor request in to the local floor request queue. As the MRFP is allowed to be treated as a conference participant or a chair, receiving the message should not be a problem, but the FCS part of the node should recognize the need for "chair-like" behavior. If there are multiple floors requested and they are handled in different MRFPs, when the floor first queued at the local FCS is in turn to be granted, the result from remote FCS is waited for if not yet received. Whenever also the remote resources are granted with ChairAction messages, the local request containing all the floors is granted similarly as for a regular case from multiple floors. If the "remote floor" is the only floor requested, the local FCS should act as in normal case with chair. Here the handling would slightly differ from the cascaded models mentioned since those use periodical FloorQuery and FloorStatus messages to attain queuing effect because of the multiple FCSs have queues for the *same* floor. In case there are concurrency requirements for local and remote floors, the floors are to be requested within a single BFCP floor request message which guarantees virtually simultaneous granting at the local FCS.

Concerning the media manipulation, the stream modification of local terminations is not enough for aforementioned circumstances, when the remote stream is handled by a remote FCS, which is possibly having participants of its own. The knowledge of what medias actually are controlled by which processors and servers might bring a problem for conference indications, for a FCS which is forcing the stream modes, for floors synchronization and in some cases on instructing the remote media mixer. The

[36] MRFP/FCS owns a floor = is responsible for mixing resources for the media the floor is dedicated to.

cascaded conferencing models introduced are symmetric in this property, but information on media ownership distribution should be kept in our distributed media server case in the common application server. A solution to this problem, instead of duplicating the information at all FCSs, is to make a *Notify* command towards the AS/MRFC *every time* a floor is granted locally in association with a "remote floor". The AS/MRFC would then know to which MRFP to direct the instructions. This does not still solve the timing issues when the MRFPs are allowed to act autonomously, e.g., granting the floor for two different medias, handled by different MRFPs, in the same time when the floor request is indicating simultaneous request. The remote FCS should notice, that since the floor request came from a different FCS and could be only a part of another floor request, it should not request to change the related stream mode before all the floors are granted. This could be even indicated in the FloorStatus (pending) message first received at the remote FCS. When all the floors were granted, the Notify from the local FCS would inform AS, which would in turn inform the remote FCS to act accordingly. Note also, that even without forced media modification for the terminations, the remote media mixer might still need instructions from AS/MRFC to adjust the streams when a combination of floors is granted in another FCS, e.g. for video input and tiling of a large conference. These notifications, instructions and the sum of participants for the joined conferences make the total traffic during the conference on respective Mp interfaces.

Multiple streams handled in different MRFPs, only one FCS. Another solution to concurrency handling of multiple MRFPs is, that there would be only one floor control server and the owner of the remote media resource would do conference mixing. When a *simple cascaded conference* model [Nov06] shown in Figure 6-8 is applied to our scenario, the floors are controlled actively only in the sole *primary* conference server, while the *secondary* conference servers relay the BFCP messages towards the *primary*.

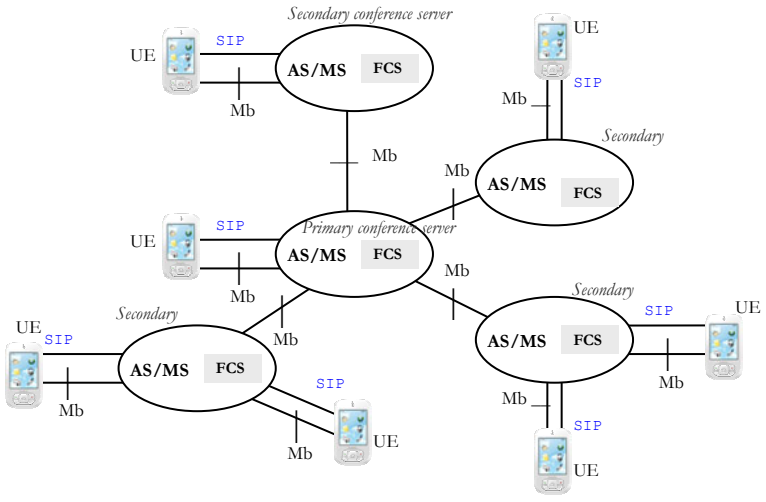

Figure 6-8 Simple cascaded conferencing model

The floor control servers locally all look like active servers, but the floor control server in the *primary* conference server sees the *secondary* conference servers as floor participants. The floor requests are not possible to be handled in secondary conference servers, and the control over possession of the floor is done completely in the *primary* conference server. In our scenario this model could be reused from mixing perspective with some modifications to floor control, see Figure 6-9 for media server nodes placement. Instead of relaying the BFCP messages from a FCS in remote MRFP, the participants would be instructed all to make the TCP connections to the local MRFP denoted as *primary*. The *primary* MRFP would have the only FCS and it would treat the remote MRFP as a *secondary* MRFP for the specific floor doing the mixing for it.

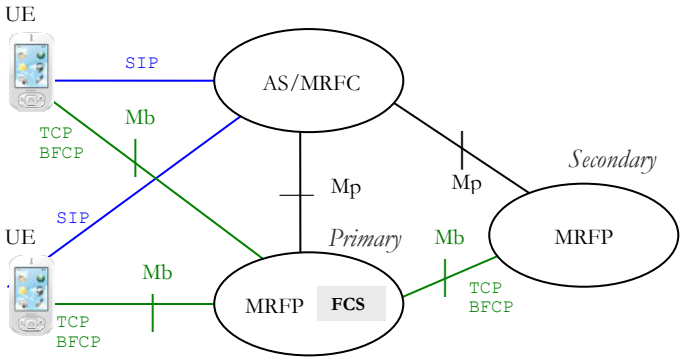

Figure 6-9 Multiple MRFPs, only one FCS in primary MRFP

Now the FCS would need to instruct the remote mixer and possibly the remote participants on what comes to changing statuses in the way described in the "multiple FCSs" option.

Distributed media handling vs. regional cascaded conferencing model. The *regional cascaded conferencing* model, regarded as the most adequate of the cascaded models [Nov06], when focusing on floor control is a blend of the previously mentioned cascaded models. The model reminds of a cluster of *primary* conference servers handling *secondary* conference servers. The cluster connections between the primary conference servers in turn act like in a *full mesh cascaded conferencing* model. Furthermore, as result the floor ownership handling should be done in the "primary conference server network" level and thus should not affect the Mp interfaces more than in the *full mesh cascaded conferencing* model. The SDP negotiation for the *primary* conference servers joining the conference is done with external means from thesis perspective and since the negotiation does not cross the Mp reference point (Mp interface), it is out of scope.

6.3 Floor control server in Multimedia Resource Function Controller

In this subsection we question the setting in the previous subsection and discuss the idea, allowed by the IETF works, what would happen if the floor control server would be located in the MRFC side of the Mp interface. In this scenario the TCP connection for BFCP traffic would be terminated in the MRFP. Figure 6-10 shows the changed network setting.

The media server work division model introduced by combined AS/MRFC/FCS brings along two major changes on Mp interface. Firstly, the floor control server does not need to use Mp interface to request media manipulation or indications and updating the conference data. Part of these were taken care of already in processor level in case of the autonomous MRFP (having FCS), nevertheless, the commands to modify the termination data are to be passed on the interface in a normal way since the floor control is not now with the MRFP. Secondly, all the floor control messages have to be relayed on Mp interface since the TCP termination is in MRFP. In addition to these, the problem with having the different medias controlled by different MRFPs is not there anymore, since the control of the distributed medias can be fluently handled from a combined AS/MRFC/FCS providing that the work division for mixing is taken care of.

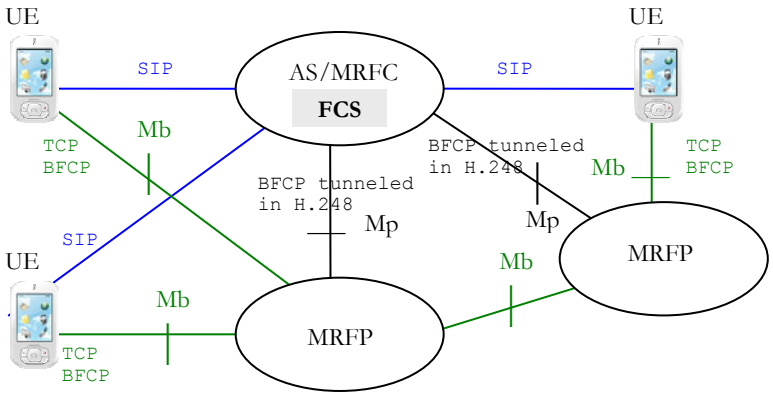

Figure 6-10 FCS in MRFC, TCP/BFCP connections terminated in MRFP

Requirements analysis and scenario. The property for basic service activation, "Floor Control" (*fc*) and the event defined for the "FCS in MRFP" situation are valid for the network scenario in this subsection. When the property indicating "floor control tunneling" (*fct*) for a termination is received in an *Add* request, the MRFP would have to be prepared to relay all the floor control messages from user terminations towards MRFC and to be expecting some floor control messages back to be translated to BFCP. In addition an event "BFCP message detection" describing the tunneled BFCP messages in ObservedEventDescriptor parameters is needed. The

event is requested to be reported from MRFP with a *Notify,* which is then piggybacking the BFCP message received in the user plane. In the opposite direction the BFCP messages would be tunneled in *Modify* requests. The tunneling procedure could be analogous to what is done when tunneling IP Bearer Control Protocol (IPBCP) in H.248 messages, accomplished by using the generic tunneling mechanism Bearer Independent Call Control (BICC) Bearer Control Tunneling Protocol [Itu01]. The H.248 messages tunneling BFCP would be relatively small comparing to typical H.248 textual or binary messages. As a setback, the tunneling messages might still multiply the traffic on Mp interface.

In this use case we assume that the floor control is taken into use in conference level. The MRFP is required to set up or accept the TCP terminations for BFCP traffic whenever the *Add* is received indicating floor control in the remote SDP and to do the media mixing as is requested by the AS/MRFC having FCS. A similar flow as in Figure 6-4 " FCS in MRFP: Conference set-up and release in Mp interface" is shown in Figure 6-11, reflecting the changes mentioned in this subsection.

The changes described do not decrease the need for notifications in most cases prior to modifications of terminations, but the floor control messages leading to the actual modification are evident, not to mention the weight to traffic brought by communication with one or multiple chairs for a conference. Even if the message load would be quite significant on Mp interface, the most of the messages and the processing is quite light - what are the processing capacity requirements for the moderate-sized and simple BFCP messages and how the parsers are to be implemented to reduce the load is another thing and out of scope of this thesis.

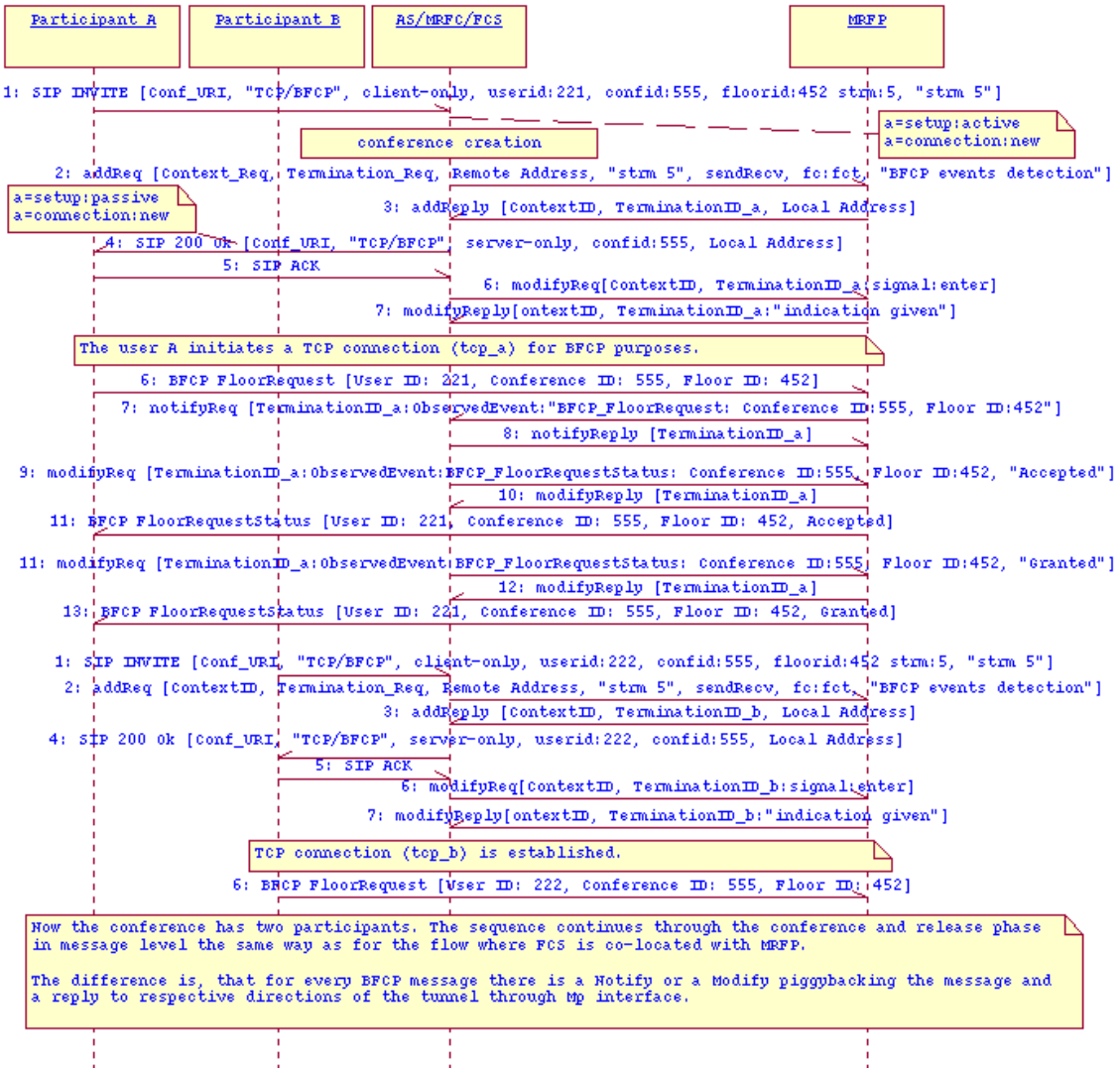

Figure 6-11 FCS in MRFC: Conference set-up and release in Mp interface

7 Conclusions

The region the thesis is exploring is in tightly coupled conferencing service area with centralized control and it uses floor control to share resources for the conference participants. Floor control comprises the features allowing the participants to utilize the conference "right to speak" concerning different conference medias, e.g. video and audio grouped into *floors*, in a coordinated manner.

The standardization bodies are in the middle of defining the conferencing service using floor control and location for the floor control server. The IETF with Internet drafts and Requests for Comments (RFC), 3G Partnership Project (3GPP) with its technical specifications and International Telecommunications Union (ITU) with its recommendations have different opinions and work in progress regarding multiple parts of the service. The thesis evaluates some of the problems. Discussions and decisions on these items are going on right now and are probably concluded before long after writing of the thesis.

The research problem of the thesis was to recommend a location for a floor control server and to evaluate the need for changes in the Mp interface emerging from floor control requirements directly or indirectly, taking into account a set of aspects influencing the choices. The two locations considered were the Multimedia Resource Function Processor (MRFP) and the combined Session Initiation Protocol (SIP) conference application server and Multimedia Resource Function Controller (MRFC). The points considered were different media streams in one conference distributed for multiple MRFPs, floor control decisions modifying the media streams and multiple floor control servers in a conference where the MRFPs would host the floor control servers. The study was accomplished with respect to H.248 [Itu05] messages and parameters giving a basic requirements analysis and a suggestion for supporting the service in the Mp interface between the MRFC and MRFP. Before the actual Mp interface was examined, the basis for understanding the challenge was established: the network environment, conferencing and protocols - Session Description Protocol (SDP) [HJP06], SIP [Ros02], Binary Floor control Protocol (BFCP) [COD06] and H.248 with their basic features and typical usage.

Both of the examined locations for the floor control server have different advantages. Placing the server on MRFP level needs more changes in the Mb interface between MRFPs and leaves some unsolved problems, referring here to the simultaneous granting of multiple floors and hence media mixing timing issues. On the other hand, when considering only the Mp interface, for a solution having the floor control server in the MRFC the amount of messages is increased or even multiplied by the floor

control messages tunneled in the H.248 messages. The fact that floor control could directly talk to the MRFC aids eliminating message exchange in certain cases. This can happen when MRFP must request media stream modifications from MRFC when floor control is set to force media manipulation, i.e., the "right to speak" is granted or taken away by shutting down or allowing the actual stream. When the floor control server is in the MRFC, the reducing request messages goes for both tone and indication requests alike. From the perspective set by this study the selection between the locations are in the end to be made according to which points are the most defective ones in relation to the floor control usage graph concerning the interface and media server performance.

In conclusion, the choice of having the floor control server in only one of the places is not the most elegant solution. The possibility to have the floor control server in MRFP or interchangeably in MRFC should be decided on conference or network basis, depending also on the profile of the service. This can be achieved by the implementation proposal given in this thesis. The standardization should take both possibilities into account and make the service consistent for both options to coexist. If one specific place for the floor control server is to be fixed, more work is to be done in connected areas to solve the unfinished issues outside scope of this thesis to discover the most suitable location, if realistic.

References

3GPP07a 3GPP TR 21.905 V8.2.0, Technical Specification Group Services and System Aspects, **Vocabulary** for 3GPP Specifications, Technical Report, Release 8. 3rd Generation Partnership Project, September 2007. Available at http://www.3gpp.org/ftp/Specs/archive/21_series/21.905/

3GPP07b 3GPP TS 23.228 V8.2.0, Technical Specification Group **Core Services and System Aspects**, IP Multimedia System (IMS), Stage 2, Release 8. 3rd Generation Partnership Project, September 2007. Available at http://www.3gpp.org/ftp/Specs/archive/23_series/23.228

3GPP07c 3GPP TS 24.147 V7.6.0, Technical Specification Group Core Network and Terminals, **Conferencing using the IP Multimedia (IM)** Core Network (CN) subsystem, Stage 3, Release 7. 3rd Generation Partnership Project, September 2007. Available at http://www.3gpp.org/ftp/Specs/archive/24_series/24.147

3GPP07d 3GPP TS 23.002 V8.1.1, Technical Specification Group Services and System Aspects, Network **architecture**, Technical Specification, Release 8. 3rd Generation Partnership Project, October 2007. Available at http://www.3gpp.org/ftp/Specs/archive/23_series/23.002

3GPP07e 3GPP TS 24.880 V1.4.0, Technical Specification Group Core Network and Terminals, **Media Control using** the IP Multimedia (**IM**) Core Network (CN) subsystem, Stage 3, Release 8. 3rd Generation Partnership Project, October 2007. Available at http://www.3gpp.org/ftp/Specs/archive/24_series/24.880

3GPP07f 3GPP TS 23.333 V7.3.0, Technical Specification Group Core Network and Terminals, Multimedia Resource Function Controller (MRFC) - Multimedia Resource Function Processor (MRFP) **Mp Interface:** Procedures descriptions, Release 7. 3rd Generation Partnership Project, December 2007. Available at http://www.3gpp.org/ftp/Specs/archive/23_series/23.333

3GPP07g 3GPP TS 29.333 V7.2.0, Technical Specification Group Core Network and Terminals, Multimedia Resource Function Controller (MRFC) - Multimedia Resource Function Processor (MRFP) **Mp Interface** – Stage 3, Release 7. 3rd Generation Partnership Project, December 2007. Available at http://www.3gpp.org/ftp/Specs/archive/29_series/29.333

3GPP08 3GPP, **23.333 CR 34.2** rev 7.3.0, C4-080509, "**Clarify floor control requirement**", Change Request to 3GPP TSG CT WG4 Meeting #38 in Puerto Vallarta, Mexico, 28 January – 1 February 2008. Available at ftp://ftp.3gpp.org/tsg_ct/WG4_protocollars_ex-CN4/TSGCT4_38_Puerto_Vallarta_2008_01/Docs/

ABW06 Andreasen, F., Baugher, M., Wing D., Session Description Protocol **(SDP) Security Descriptions for Media Streams**, RFC **4568**. Internet Society, July 2006. http://www.ietf.org/rfc/rfc4568.txt

Ark06 Arkko, J., et al., **Key Management Extensions for** Session Description Protocol (**SDP**) and Real Time Streaming Protocol (**RTSP**), RFC **4567**. Internet Society, July 2006. Available at http://www.ietf.org/rfc/rfc4567.txt

Bal93 Balenson, D., Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers, RFC **1423**. Internet Society, February 1993. Available at http://www.ietf.org/rfc/rfc1423.txt

BBL07 Barnes, M., Boulton, C., Levin, O., A **Framework** for Centralized Conferencing, work in progress. Internet Society, November 9, 2007. Expires: May 12, 2008. Available at http://www.ietf.org/internet-drafts/draft-ietf-xcon-framework-10.txt [14.11.2007]

Ber05 Berners-Lee, T. et al., Uniform Resource Identifiers (**URI**): Generic Syntax, RFC 3986. Internet Society, January 2005. http://www.ietf.org/rfc/rfc3986.txt

Buo07 Buono, A., et al., A Distributed IMS Enabled Conferencing Architecture on Top of a Standard Centralized Conferencing Framework, *IEEE Communications Magazine*, March 2007, pages 152-159.

CaG05 Camarillo, G., Garcia-Martin, M. A., The 3G IP Multimedia Subsystem – Merging the Internet and the Cellular Worlds. Wiley 2004.

Cam02 Camarillo, G., SIP Demystified. McGraw-Hill, 2002.

Cam06a Camarillo, G., Binary Floor Control Protocol (BFCP), presentation, 2006. Available at http://www.netlab.tkk.fi/opetus/s383115/2006/kalvot/gc-bfcp.pdf [13.12.2007]

Cam06b Camarillo, G., Session Description Protocol (**SDP**) Format for Binary Floor Control Protocol (**BFCP**) Streams, RFC **4583**. Internet Society, September 2007. Available at http://www.ietf.org/rfc/rfc4583.txt

Cam07 Camarillo, G., **Connection Establishment** in the Binary Floor Control Protocol (BFCP), RFC **5018**. Internet Society, September 2007. http://www.ietf.org/rfc/rfc5018.txt

COD06 Camarillo, G., Ott, J., Drage, K., The Binary Floor Control Protocol (**BFCP**), RFC **4582**. Internet Society, November 2006. Available at http://www.ietf.org/rfc/rfc4582.txt

DiR06 Dierks, T., E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.1, RFC **4346**. Internet Society, April 2006. Available at http://www.ietf.org/rfc/rfc4346.txt

Eri07a Ericsson AB, Introduction to IMS, 284 23-8123 Uen Rev A, White Paper, March 2007. Available at http://www.ericsson.com/technology/whitepapers/8123_Intro_to_ims_a.pdf [14.12.2007]

Eri07b Ericsson AB, Services in the IMS ecosystem, 285 23-3109 Uen Rev A, White Paper, February 2007. Available at http://www.ericsson.com/technology/whitepapers/3109 Services in the IMS ecosystem A.pdf [14.12.2007]

EvI06 Even, R., Ismail, N., **Conferencing Scenarios**, RFC **4597**. Internet Society, July 2006. Available at http://www.ietf.org/rfc/rfc4597.txt

Fie99 Fielding R. et al., Hypertext Transfer Protocol – **HTTP**/1.1, RFC **2616**. Internet Society, June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt

Fra99 Franks, J. et al., HTTP **authentication**: Basic and Digest Access Authentication, RFC **2617**. Internet Society, June 1999. Available at http://www.ietf.org/rfc/rfc2617.txt

FrB96 Freed, N., Borenstein, N., Multipurpose Internet Mail Extensions (**MIME**) Part One: Format of Internet Message Bodies, RFC **2045**. Internet Society, November 1996. Available at http://www.ietf.org/rfc/rfc2045.txt

Gal95 Galvin, J. et al., **Security** Multiparts for **MIME**: Multipart/Signed and Multipart/Encrypted, RFC **1847**. Internet Society, October 1995. Available at http://www.ietf.org/rfc/rfc1847.txt

HJP06 Handley, M., Jacobson, V., Perkins, C., **SDP**: Session Description Protocol, RFC **4566**. Internet Society, July 2006. Available at http://www.ietf.org/rfc/rfc4566.txt

HPW00 Handley, M., Perkins, C., Whelan, E., Session Announcement Protocol (**SAP**), RFC **2974**. Internet Society, October 2000. Available at http://www.ietf.org/rfc/rfc2974.txt

Itu01 ITU-T, ITU-T Recommendation **Q.1990**. Series Q: Switching and Signaling, Specifications of signaling related to Bearer Independent Call Control (BICC), BICC bearer control tunneling protocol, International Telecommunications Union (ITU-T), July 2001. http://www.itu.int/rec/T-REC-Q.1990/en

Itu03 ITU-T, ITU-T Recommendation **H.248.27**. Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services – Communication procedures, Gateway control protocol: **Supplemental tones packages**, International Telecommunications Union (ITU-T), July 2003. http://www.itu.int/rec/T-REC-H.248.27/en

Itu04 ITU-T, ITU-T Recommendation **H.248.19**. Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services – Communication procedures, Gateway control protocol: **Decomposed multipoint control unit, audio, video and data conferencing packages**, International Telecommunications Union (ITU-T), March 2004. http://www.itu.int/rec/T-REC-H.248.19/en

Itu05 ITU-T, ITU-T Recommendation **H.248.1**. Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services – Communication procedures, Gateway control protocol: **Version 3**, International Telecommunications Union (ITU-T), September 2005. http://www.itu.int/rec/T-REC-H.248.1/en

Itu08 ITU-T, AVD-3332: **H.248.19 Amendment 2** Editor's Input, Proposal, Telecommunication Standardization Sector, International Telecommunications Union (ITU-T), 17-23 January 2008. Available at http://ftp3.itu.ch/av-arch/avc-site/2005-2008/0801_Seo/

JoL06 Johnston, A., O. Levin, Session Initiation Protocol (SIP) Call Control – **Conferencing for User Agents**, RFC **4579**. Internet Society, August 2006. Available at http://www.ietf.org/rfc/rfc4579.txt

KeS05 Kent, S., R. Atkinson, **Security Architecture** for the Internet Protocol, RFC **4301**. Internet Society, December 2005. Available at http://www.ietf.org/rfc/rfc4301.txt

KKS01 Kumar, V., Korpi, M., Sengodan, S., *IP Telephony with H.323*. Wiley, 2001, 347-370.

Kos06 Koskelainen P. et. Al., **Requirements** for Floor Control Protocols, RFC **4376**. Internet Society, February 2006. Available at http://www.ietf.org/rfc/rfc4376.txt

LeC06 Levin, O., Camarillo, G., The Session Description Protocol (**SDP**) Label Attribute, RFC **4574**. Internet Society, August 2006. Available at http://www.ietf.org/rfc/rfc4574.txt

Len06 Lennox, J., Connection-Oriented Media Transport over the Transport Layer Security (**TLS**) Protocol in the Session Description Protocol (**SDP**), RFC **4572**. Internet Society, July 2006. Available at http://www.ietf.org/rfc/rfc4572.txt

LeS03 Lennox, J., Schulzrinne, H., A Protocol For Reliable Decentralized Conferencing, *ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Monterrey, California, June 2003. Available at http://www1.cs.columbia.edu/~lennox/

Nov06 Novo, O., Study of Cascaded Conferences and Implementation of the Binary Floor Control Protocol (BFCP) for Centralized Conferences, Master's Thesis. Helsinki University of Technology, 2006. Available at http://www.tml.tkk.fi/~anttiyj/Oscar-BFCP.pdf

Nov07 Novo, O. et. Al., Conference Information **Data Model for Centralized Conferencing (XCON)**, work in progress. Internet Society, October 31, 2007. Expires: May 3, 2008. Available at http://tools.ietf.org/html/draft-ietf-xcon-common-data-model-06.txt [1.11.2007]

Pac08 Packetizer, H.248 Information Site, 2008. http://www.packetizer.com/voip/h248/ [14.1.2008]

Que05 Quemada, J., et al., Isabel: An Application for Real Time Collaboration with a Flexible Floor Control, December 5, San Jose, USA. http://isabel.dit.upm.es

Ram04 Ramsdell B., Secure/Multipurpose Internet Mail Extensions (**S/MIME**) Version 3.1 Message Specification, RFC **3851**. Internet Society, July 2004. Available at http://www.ietf.org/rfc/rfc3851.txt

Ram08 Ramos, E. J., **Analyzing the Media Control interfaces** and Mobile Media Gateway for IP Multimedia Subsystems (IMS), Master's Thesis. Helsinki University of Technology, 2008.

Res01 Resnick, P., Internet Message Format, RFC **2822**. Internet Society, April 2001. Available at http://www.ietf.org/rfc/rfc2822.txt

Roa02 Roach, A., Session Initiation Protocol **(SIP)-Specific Event Notification**, RFC **3265**. Internet Society, June 2002. Available at http://www.ietf.org/rfc/rfc3265.txt

RoS02 Rosenberg, J. and Schulzrinne, H., An **Offer/Answer** Model with the Session Description Protocol (**SDP**), RFC **3264**. Internet Society, June 2002. Available at http://www.ietf.org/rfc/rfc3264.txt

Ros02 Rosenberg, J. et al., **SIP**: Session Initiation Protocol, RFC **3261**. Internet Society, June 2002. Available at http://www.ietf.org/rfc/rfc3261.txt

RoS06 Rosenberg, J. and Schulzrinne, H., **Guidelines** for Authors of Extensions to the Session Initiation Protocol (SIP), RFC **4485**. Internet Society, May 2006. Available at http://www.ietf.org/rfc/rfc4485.txt

Ros06 Rosenberg, J., A Framework for **Conferencing** with the Session Initiation Protocol (**SIP**), RFC **4353**. Internet Society, February 2006. Available at http://www.ietf.org/rfc/rfc4353.txt

RSL06 Rosenberg, J., Schulzrinne, H. and Levin O., A Session Initiation Protocol **(SIP) Event Package for Conference State**, RFC **4575**. Internet Society, August 2006. Available at http://www.ietf.org/rfc/rfc4575.txt

SAH07 Sandbakken, G. (ed.), Andersen, T., Heggestad, A., Binary Floor Control Protocol over **UDP**, work in progress. Internet Society, October 10, 2007. Available at http://tools.ietf.org/html/draft-**sandbakken**-xcon-bfcp-udp-00 [1.11.2007]

Sch03 Schulzrinne, H., et al., RTP: A Transport for Real-Time Applications, RFC **3550**. Internet Society, July 2003. Available at http://www.ietf.org/rfc/rfc3550.txt

SRL98 Schulzrinne, H., Rao, A., Lanphier, R., Real Time Streaming Protocol (**RTSP**), RFC **2326**. Internet Society, April 1998. Available at http://www.ietf.org/rfc/rfc2326.txt

YoC05 Yon, D., Camarillo, G., **TCP-Based Media Transport** in the Session Description Protocol (**SDP**), RFC **4145**. Internet Society, September 2005. Available at http://www.ietf.org/rfc/rfc4145.txt

Appendix A. Protocol Formats, Tables and Examples

A.1 SDP ABNF Grammar

The current ABNF grammar for the SDP is introduced in the protocol specification [HJP06], here listed below.

```
; SDP Syntax
session-description = proto-version
                      origin-field
                      session-name-field
                      information-field
                      uri-field
                      email-fields
                      phone-fields
                      connection-field
                      bandwidth-fields
                      time-fields
                      key-field
                      attribute-fields
                      media-descriptions

proto-version =       %x76 "=" 1*DIGIT CRLF
                      ;this memo describes version 0

origin-field =        %x6f "=" username SP sess-id SP sess-version SP
                      nettype SP addrtype SP unicast-address CRLF

session-name-field =  %x73 "=" text CRLF

information-field =   [%x69 "=" text CRLF]

uri-field =           [%x75 "=" uri CRLF]

email-fields =        *(%x65 "=" email-address CRLF)

phone-fields =        *(%x70 "=" phone-number CRLF)

connection-field =    [%x63 "=" nettype SP addrtype SP
                      connection-address CRLF]
                      ;a connection field must be present
                      ;in every media description or at the
                      ;session-level

bandwidth-fields =    *(%x62 "=" bwtype ":" bandwidth CRLF)

time-fields =         1*( %x74 "=" start-time SP stop-time
                      *(CRLF repeat-fields) CRLF)
                      [zone-adjustments CRLF]

repeat-fields =       %x72 "=" repeat-interval SP typed-time
                      1*(SP typed-time)

zone-adjustments =    %x7a "=" time SP ["-"] typed-time
                      *(SP time SP ["-"] typed-time)

key-field =           [%x6b "=" key-type CRLF]

attribute-fields =    *(%x61 "=" attribute CRLF)

media-descriptions =  *( media-field
                      information-field
                      *connection-field
                      bandwidth-fields
                      key-field
                      attribute-fields )
```

```
media-field =          %x6d "=" media SP port ["/" integer]
                       SP proto 1*(SP fmt) CRLF

; sub-rules of 'o='
username =             non-ws-string
                       ;pretty wide definition, but doesn't
                       ;include space

sess-id =              1*DIGIT
                       ;should be unique for this username/host

sess-version =         1*DIGIT

nettype =              token
                       ;typically "IN"

addrtype =             token
                       ;typically "IP4" or "IP6"

; sub-rules of 'u='
uri =                  URI-reference
                       ; see RFC 3986

; sub-rules of 'e=', see RFC 2822 for definitions
email-address        = address-and-comment / dispname-and-address
                         / addr-spec
address-and-comment  = addr-spec 1*SP "(" 1*email-safe ")"
dispname-and-address = 1*email-safe 1*SP "<" addr-spec ">"

; sub-rules of 'p='
phone-number =         phone *SP "(" 1*email-safe ")" /
                       1*email-safe "<" phone ">" /
                       phone

phone =                ["+"] DIGIT 1*(SP / "-" / DIGIT)

; sub-rules of 'c='
connection-address =  multicast-address / unicast-address

; sub-rules of 'b='
bwtype =               token

bandwidth =            1*DIGIT

; sub-rules of 't='
start-time =           time / "0"

stop-time =            time / "0"

time =                 POS-DIGIT 9*DIGIT
                       ; Decimal representation of NTP time in
                       ; seconds since 1900.  The representation
                       ; of NTP time is an unbounded length field
                       ; containing at least 10 digits.  Unlike the
                       ; 64-bit representation used elsewhere, time
                       ; in SDP does not wrap in the year 2036.

; sub-rules of 'r=' and 'z='
repeat-interval =      POS-DIGIT *DIGIT [fixed-len-time-unit]

typed-time =           1*DIGIT [fixed-len-time-unit]

fixed-len-time-unit = %x64 / %x68 / %x6d / %x73
```

```
; sub-rules of 'k='
key-type =            %x70 %x72 %x6f %x6d %x70 %x74 /     ; "prompt"
                      %x63 %x6c %x65 %x61 %x72 ":" text / ; "clear:"
                      %x62 %x61 %x73 %x65 "64:" base64 /  ; "base64:"
                      %x75 %x72 %x69 ":" uri              ; "uri:"

base64      =         *base64-unit [base64-pad]

base64-unit =         4base64-char
base64-pad  =         2base64-char "==" / 3base64-char "="
base64-char =         ALPHA / DIGIT / "+" / "/"

; sub-rules of 'a='
attribute =           (att-field ":" att-value) / att-field

att-field =           token

att-value =           byte-string

; sub-rules of 'm='
media =               token
                      ;typically "audio", "video", "text", or
                      ;"application"

fmt =                 token
                      ;typically an RTP payload type for audio
                      ;and video media

proto  =              token *("/" token)
                      ;typically "RTP/AVP" or "udp"

port =                1*DIGIT

; generic sub-rules: addressing
unicast-address =     IP4-address / IP6-address / FQDN / extn-addr

multicast-address =   IP4-multicast / IP6-multicast / FQDN
                      / extn-addr

IP4-multicast =       m1 3( "." decimal-uchar )
                      "/" ttl [ "/" integer ]
                      ; IPv4 multicast addresses may be in the
                      ; range 224.0.0.0 to 239.255.255.255

m1 =                  ("22" ("4"/"5"/"6"/"7"/"8"/"9")) /
                      ("23" DIGIT )

IP6-multicast =       hexpart [ "/" integer ]
                      ; IPv6 address starting with FF

ttl =                 (POS-DIGIT *2DIGIT) / "0"

FQDN =                4*(alpha-numeric / "-" / ".")
                      ; fully qualified domain name as specified
                      ; in RFC 1035 (and updates)

IP4-address =         b1 3("." decimal-uchar)

b1 =                  decimal-uchar
                      ; less than "224"

; The following is consistent with RFC 2373 [30], Appendix B.
IP6-address =         hexpart [ ":" IP4-address ]

hexpart =             hexseq / hexseq "::" [ hexseq ] /
                      "::" [ hexseq ]

hexseq  =             hex4 *( ":" hex4)

hex4    =             1*4HEXDIG

; Generic for other address families
extn-addr =           non-ws-string
```

```
; generic sub-rules: datatypes
text =                byte-string
                      ;default is to interpret this as UTF8 text.
                      ;ISO 8859-1 requires "a=charset:ISO-8859-1"
                      ;session-level attribute to be used

byte-string =         1*(%x01-09/%x0B-0C/%x0E-FF)
                      ;any byte except NUL, CR, or LF

non-ws-string =       1*(VCHAR/%x80-FF)
                      ;string of visible characters

token-char =          %x21 / %x23-27 / %x2A-2B / %x2D-2E / %x30-39
                      / %x41-5A / %x5E-7E

token =               1*(token-char)

email-safe =          %x01-09/%x0B-0C/%x0E-27/%x2A-3B/%x3D/%x3F-FF
                      ;any byte except NUL, CR, LF, or the quoting
                      ;characters ()<>

integer =             POS-DIGIT *DIGIT

; generic sub-rules: primitives
alpha-numeric =       ALPHA / DIGIT

POS-DIGIT =           %x31-39 ; 1 - 9

decimal-uchar =       DIGIT
                      / POS-DIGIT DIGIT
                      / ("1" 2*(DIGIT))
                      / ("2" ("0"/"1"/"2"/"3"/"4") DIGIT)
                      / ("2" "5" ("0"/"1"/"2"/"3"/"4"/"5"))

; external references:
      ; ALPHA, DIGIT, CRLF, SP, VCHAR: from RFC 4234
      ; URI-reference: from RFC 3986
      ; addr-spec: from RFC 2822
```

A.2 SIP Header Fields

Information about SIP header fields in relation to methods and proxy processing is summarized in Table A.1 and Table A.2.

The "**where**" column describes the request and response types in which the header field can be used. Values in this column are:

R header field may only appear in requests;
r header field may only appear in responses;
2xx, 4xx, etc**.**
A numerical value or range indicates response codes with which the header field can be used;
c: header field is copied from the request to the response.
An empty entry
in the "where" column indicates that the header field may be present in all requests and responses.

The "**proxy**" column describes the operations a proxy may perform on a header field:

a: A proxy can add or concatenate the header field if not present.
m: A proxy can modify an existing header field value.
d: A proxy can delete a header field value.
r: A proxy must be able to read the header field, and thus this header field cannot be encrypted.

The next six columns relate to the presence of a header field in a method:

c Conditional; requirements on the header field depend on the context of the message.
m The header field is mandatory.
m* The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field.
o The header field is optional.
t The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field. If a stream-based protocol (such as TCP) is used as a transport, then the header field MUST be sent.
***** The header field is required if the message body is not empty. See [Ros02a, chap 20.14, chap 20.15, chap 7.4] for details.
- The header field is not applicable.

"**Optional**" means that an element MAY include the header field in a request or response, and a UA MAY ignore the header field if present in the request or response (The exception to this rule is the Require header field discussed in [Ros02a, chap 20.32]). A "**mandatory**" header field MUST be present in a request, and MUST be understood by the UAS receiving the request. A mandatory response header field MUST be present in the response, and the header field MUST be understood by the UAC processing the response. "**Not applicable**" means that the header field MUST NOT be present in a request. If one is placed in a request by mistake, it MUST be ignored by the UAS receiving the request. Similarly, a header field labelled "not applicable" for a response means that the UAS MUST NOT place the header field in the response, and the UAC MUST ignore the header field in the response.

A UA SHOULD ignore extension header parameters that are not understood.

A compact form of some common header field names is also defined for use when overall message size is an issue. [Ros02a, pp160]

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept	R		-	o	-	o	m*	o
Accept	2xx		-	-	-	o	m*	o
Accept	415		-	c	-	c	c	c
Accept-Encoding	R		-	o	-	o	o	o
Accept-Encoding	2xx		-	-	-	o	m*	o
Accept-Encoding	415		-	c	-	c	c	c
Accept-Language	R		-	o	-	o	o	o
Accept-Language	2xx		-	-	-	o	m*	o
Accept-Language	415		-	c	-	c	c	c
Alert-Info	R	ar	-	-	-	o	-	-
Alert-Info	180	ar	-	-	-	o	-	-
Allow	R		-	o	-	o	o	o
Allow	2xx		-	o	-	m*	m*	o
Allow	r		-	o	-	o	o	o
Allow	405		-	m	-	m	m	m
Authentication-Info	2xx		-	o	-	o	o	o
Authorization	R		o	o	o	o	o	o
Call-ID	c	r	m	m	m	m	m	m
Call-Info		ar	-	-	-	o	o	o
Contact	R		o	-	-	m	o	o
Contact	1xx		-	-	-	o	-	-
Contact	2xx		-	-	-	m	o	o
Contact	3xx	d	-	o	-	o	o	o
Contact	485		-	o	-	o	o	o
Content-Disposition			o	o	-	o	o	o
Content-Encoding			o	o	-	o	o	o
Content-Language			o	o	-	o	o	o
Content-Length		ar	t	t	t	t	t	t
Content-Type			*	*	-	*	*	*
CSeq	c	r	m	m	m	m	m	m
Date		a	o	o	o	o	o	o
Error-Info	300-699	a	-	o	o	o	o	o
Expires			-	-	-	o	-	o
From	c	r	m	m	m	m	m	m
In-Reply-To	R		-	-	-	o	-	-
Max-Forwards	R	amr	m	m	m	m	m	m
Min-Expires	423		-	-	-	-	-	m
MIME-Version			o	o	-	o	o	o
Organization		ar	-	-	-	o	o	o

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Priority	R	ar	-	-	-	o	-	-
Proxy-Authenticate	407	ar	-	m	-	m	m	m
Proxy-Authenticate	401	ar	-	o	o	o	o	o
Proxy-Authorization	R	dr	o	o	-	o	o	o
Proxy-Require	R	ar	-	o	-	o	o	o
Record-Route	R	ar	o	o	o	o	o	-
Record-Route	2xx,18x	mr	-	o	o	o	o	-
Reply-To			-	-	-	o	-	-
Require		ar	-	c	-	c	c	c
Retry-After	404,413,480,486		-	o	o	o	o	o
	500,503		-	o	o	o	o	o
	600,603		-	o	o	o	o	o
Route	R	adr	c	c	c	c	c	c
Server	r		-	o	o	o	o	o
Subject	R		-	-	-	o	-	-
Supported	R		-	o	o	m*	o	o
Supported	2xx		-	o	o	m*	m*	o
Timestamp			o	o	o	o	o	o
To	c(1)	r	m	m	m	m	m	m
Unsupported	420		-	m	-	m	m	m
User-Agent			o	o	o	o	o	o
Via	R	amr	m	m	m	m	m	m
Via	rc	dr	m	m	m	m	m	m
Warning	r		-	o	o	o	o	o
WWW-Authenticate	401	ar	-	m	-	m	m	m
WWW-Authenticate	407	ar	-	o	-	o	o	o

Table A.2 Summary of header fields, P--Z; (1): copied with possible addition of tag `Accept: application/sdp;level=1, application/x-private, text/html` [Ros02a, pp162]

A.3 SIP Response Codes

The response codes are consistent with, and extend, HTTP/1.1 response codes. Not all HTTP/1.1 response codes are appropriate, and only those that are appropriate are given here. Other HTTP/1.1 response codes SHOULD NOT be used. Also, SIP defines a new class, 6xx. [Ros021a, chap. 21]

A.3.1 Provisional 1xx

Provisional responses, also known as informational responses, indicate that the server contacted is performing some further action and does not yet have a definitive response. A server sends a 1xx response if it expects to take more than 200 ms to obtain a final response. Note that 1xx responses are not transmitted reliably. They never cause the client to send an ACK. Provisional (1xx) responses MAY contain message bodies, including session descriptions. [Ros021a, chap. 21]

Code	Descriptive phrase	Subsection for reference in [Ros02a]
100	Trying	21.1.1
180	Ringing	21.1.2
181	Call Is Being Forwarded	21.1.3
182	Queued	21.1.4
183	Session Progress	21.1.5

Table A.3 SIP Response Codes, Provisional 1xx

A.3.2 Successful 2xx

The request was successful. [Ros021a, chap. 21]

Code	Descriptive phrase	Subsection for reference in [Ros02a]
200	OK	21.2.1

Table A.4 SIP Response Codes, Successful 2xx

A.3.3 Redirection 3xx

3xx responses give information about the user's new location, or about alternative services that might be able to satisfy the call. [Ros021a, chap. 21]

Code	Descriptive phrase	Subsection for reference in [Ros02a]
300	Multiple Choices	21.3.1
301	Moved Permanently	21.3.2
302	Moved Temporarily	21.3.3
305	Use Proxy	21.3.4
380	Alternative Service	21.3.5

Table A.5 SIP Response Codes, Redirection 3xx

A.3.4 Request Failure 4xx

4xx responses are definite failure responses from a particular server. The client SHOULD NOT retry the same request without modification (for example, adding appropriate authorization). However, the same request to a different server might be successful. [Ros021a, chap. 21]

Code	Descriptive phrase	Subsection for reference in [Ros02a]
400	Bad Request	21.4.1
401	Unauthorized	21.4.2
402	Payment Required (Reserved for future use)	21.4.3
403	Forbidden	21.4.4
404	Not Found	21.4.5
405	Method Not Allowed	21.4.6
406	Not Acceptable	21.4.7
407	Proxy Authentication Required	21.4.8
408	Request Timeout	21.4.9
410	Gone	21.4.10
413	Request Entity Too Large	21.4.11
414	Request-URI Too Long	21.4.12
415	Unsupported Media Type	21.4.13
416	Unsupported URI Scheme	21.4.14
420	Bad Extension	21.4.15
421	Extension Required	21.4.16
423	Interval Too Brief	21.4.17
480	Temporarily Unavailable	21.4.18
481	Call/Transaction Does Not Exist	21.4.19
482	Loop Detected	21.4.20
483	Too Many Hops	21.4.21
484	Address Incomplete	21.4.22
485	Ambiguous	21.4.23
486	Busy Here	21.4.24
487	Request Terminated	21.4.25
488	Not Acceptable Here	21.4.26
491	Request Pending	21.4.27
493	Undecipherable	21.4.28

Table A.6 Response Codes, Request Failure 4xx

A.3.5 Server Failure 5xx

5xx responses are failure responses given when a server itself has erred. [Ros021a, chap. 21]

Code	Descriptive phrase	Subsection for reference in [Ros02a]
500	Server Internal Error	21.5.1
501	Not Implemented	21.5.2
502	Bad Gateway	21.5.3
503	Service Unavailable	21.5.4
504	Server Time-out	21.5.5
505	Version Not Supported	21.5.6
513	Message Too Large	21.5.7

Table A.7 SIP Response Codes, Server Failure 5xx

A.3.6 Global Failures 6xx

6xx responses indicate that a server has definitive information about a particular user, not just the particular instance indicated in the Request-URI. [Ros021a, chap. 21]

Code	Descriptive phrase	Subsection for reference in [Ros02a]
600	Busy Everywhere	21.6.1
603	Decline	21.6.2
604	Does Not Exist Anywhere	21.6.3
606	Not Acceptable	21.6.4

Table A.8 SIP Response Codes, Global Failure 6xx

A.4 BFCP Attributes

The formats of the 18 attributes are given with the explanations. The type number is given in parentheses after the attribute name.

A.4.1 BENEFICIARY-ID (1), FLOOR-ID (2), FLOOR-REQUEST-ID (3)

These attributes have same field sizes but differ from each other with type and interpretation of the identifier field. The types are binary 0000001, 0000010 and 0000011 respectively, whereas the identifiers are 16-bit unsigned integer.

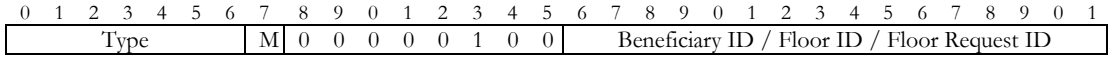

Figure A.1 BFCP identifier attribute format

The beneficiary and floor identifiers uniquely name a user and floor in respect to a specific conference. The *Beneficiary ID* is used in 3rd party floor requests and to requests information about a participant. The floor request identifier uniquely identifies a floor request in a floor control server.

A.4.2 PRIORITY (4)

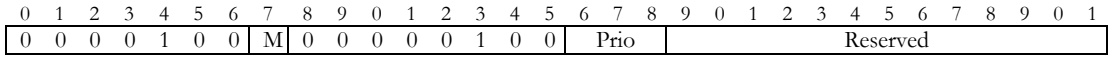

Figure A.2 BFCP priority attribute format

The there are five categories defined for priority purposes for a floor request. A value that is out of range falls back to the value *Highest* although this property should not be utilized. The default value in a message is *Normal* for a missing priority attribute. The category ranking is: *Lowest(0), Low, Normal, High, Highest(4).*

A.4.3 REQUEST-STATUS (5)

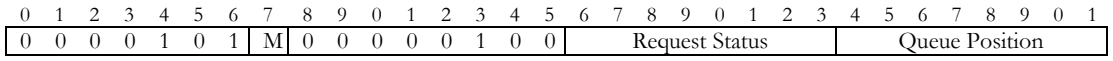

Figure A.3 BFCP request status attribute format

The there are seven stages a live floor request can be in. The status ranking is *Pending(1), Accepted, Granted, Denied, Cancelled, Released* and *Revoked(7).* The "Pending" status indicates that the floor control server has not yet received an acceptance of a request from the floor chair, in case that was required. The queue position in the implemented request queue at the server is gives when feasible.

A.4.4 **ERROR-CODE** (6)

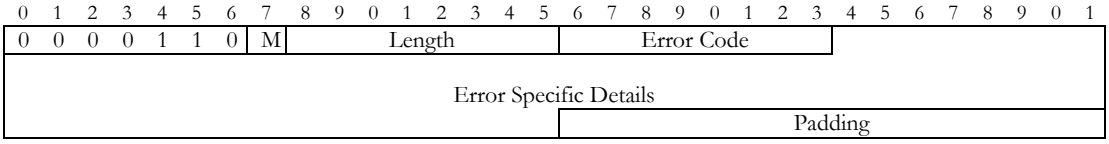

Figure A.4 BFCP error code attribute format

In case the error code is not recognized, the assumption of an error occurred is still valid but unclear and processing must continue. The error code has nine values defined:

Value	*Description*
1	Conference does not Exist
2	User does not Exist
3	Unknown Primitive
4	Unknown Mandatory Attribute
5	Unauthorized Operation
6	Invalid Floor ID
7	Floor Request ID Does Not Exist
8	You have Already Reached the Maximum Number of Floor Requests for this Floor
9	Use TLS

Table A.9 BFCP error codes

The padding (zero-bits) in the attribute can be omitted and depends on the length of the error specific details so, that the attribute will be aligned to 32-bit length. The error specific details are present in the core BFCP version 1 only for the error code *Unknown Mandatory Attribute,* format can be seen in BFCP specification [COD06].

A.4.5 **ERROR-INFO** (7)**, PARTICIPANT-PROVIDED-INFO** (8)**, STATUS-INFO** (9)**, USER-DISPLAY-NAME** (12)**, USER-URI** (13)

These attributes differ from each other with type, interpretation and length of the text field and possibly length of the padding. The types are binary 0000111, 0001000, 0001001, 0001100 and 0001101 respectively, whereas the text is a variable length UTF-8 encoded octet string. The format of the attribute is:

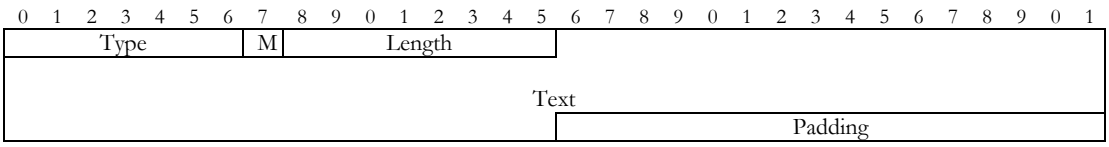

Figure A.5 BFCP common format for attributes *ERROR-INFO, PARTICIPANT-PROVIDED-INFO, STATUS-INFO, USER-DISPLAY-NAME* and *USER-URI*

The *USER-URI* attribute contains the URI which the endpoint uses to set up the streams and the BFCP connection. I our case it would be SIP URI of the user.

A.4.6 SUPPORTED-ATTRIBUTES (10), SUPPORTED-PRIMITIVES (11)

The two attributes differ with type, interpretation and length of the text field and possibly length of the padding. The types are binary 0001010 and 0001011. The text is a variable length UTF-8 encoded octet string, constructed of multiple BFCP attribute types or BFCP messages. The format of the attribute is:

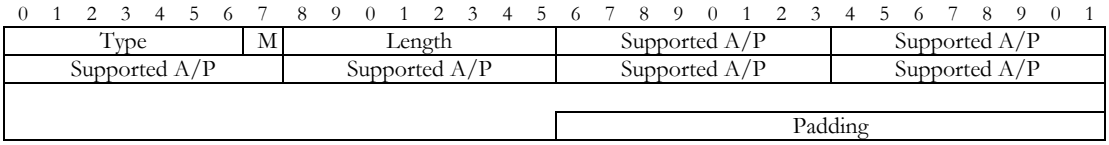

Figure A.6 BFCP format for attributes *SUPPORTED-ATTRIBUTES* and *SUPPORTED-PRIMITIVES*

The 8-bit field indicating an attribute name consists of 7-bit name and the 8:th bit is reserved for future use.

A.4.7 Grouped Attributes BENEFICIARY-INFORMATION (14), FLOOR-REQUEST-INFORMATION (15), REQUESTED-BY-INFORMATION (16), FLOOR-REQUEST-STATUS (17) and OVERALL-REQUEST-STATUS (18)

All these attributes have an own header followed by a sequence of other attributes. The header subfields are alike with one deviation - the identifier might be different. The semantics of the identifier fields are as explained earlier. In following paragraphs are formats for the headers and the ABNF forms for the corresponding attributes[37].

BENEFICIARY-INFORMATION

Header:

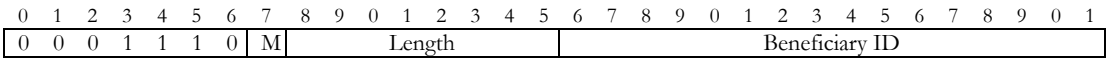

Figure A.7 BFCP *BENEFICIARY-INFORMATION* attribute header

ABNF:

BENEFICIARY-INFORMATION = (BENEFICIARY-INFORMATION-HEADER)
[USER-DISPLAY-NAME]
[USER-URI]
*[EXTENSION-ATTRIBUTE]

[37] Note that the EXTENSION-ATTRIBUTEnot mentioned elsewhere refers to future extensions to BFCP protocol attributes.

FLOOR-REQUEST-INFORMATION

Header:

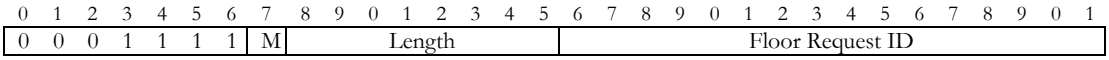

Figure A.8 BFCP *FLOOR-REQUEST-INFORMATION* attribute header

ABNF:

```
FLOOR-REQUEST-INFORMATION =   (FLOOR-REQUEST-INFORMATION-HEADER)
                              1*(FLOOR-REQUEST-STATUS)
                              [BENEFICIARY-INFORMATION]
                              [REQUESTED-BY-INFORMATION]
                              [PRIORITY]
                              [PARTICIPANT-PROVIDED-INFO]
                              *[EXTENSION-ATTRIBUTE]
```

REQUESTRD-BY-INFORMATION

Header:

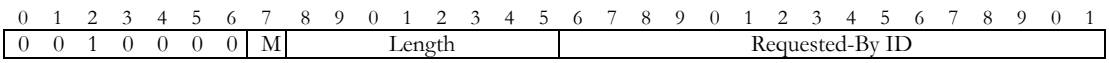

Figure A.9 BFCP *REQUESTRD-BY-INFORMATION* attribute header

The *Requested-By ID* not mentioned before is like the others identifiers – it uniquely identifies a user within a conference but is used in a different context.

ABNF:

```
REQUESTED-BY-INFORMATION = (REQUESTED-BY-INFORMATION-HEADER)
                           [USER-DISPLAY-NAME]
                           [USER-URI]
                           *[EXTENSION-ATTRIBUTE]
```

FLOOR-REQUEST-STATUS

Header:

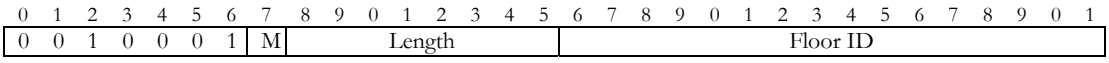

Figure A.10 BFCP *FLOOR-REQUEST-STATUS* attribute header

ABNF:

```
FLOOR-REQUEST-STATUS =    (FLOOR-REQUEST-STATUS-HEADER)
                          [REQUEST-STATUS]
                          [STATUS-INFO]
```

*[EXTENSION-ATTRIBUTE]

OVERALL-REQUEST-STATUS

Header:

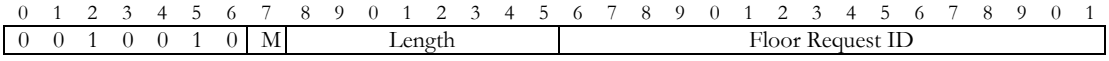

Figure A.11 BFCP *OVERALL-REQUEST-STATUS* attribute header

ABNF:

```
OVERALL-REQUEST-STATUS  =  (OVERALL-REQUEST-STATUS-HEADER)
                           [REQUEST-STATUS]
                           [STATUS-INFO]
                           *[EXTENSION-ATTRIBUTE]
```

A.5 ABNF forms of BFCP Messages

The ABNF forms with header and attributes[38] for the 13 messages are given in next subsections. The primitives are listed in type numbering order.

A.5.1 FloorRequest

```
FloorRequest =      (COMMON-HEADER)
                  1*(FLOOR-ID)
                    [BENEFICIARY-ID]
                    [PARTICIPANT-PROVIDED-INFO]
                    [PRIORITY]
                   *[EXTENSION-ATTRIBUTE]
```

A.5.2 FloorRelease

```
FloorRelease =      (COMMON-HEADER)
                    (FLOOR-REQUEST-ID)
                   *[EXTENSION-ATTRIBUTE]
```

A.5.3 FloorRequestQuery

```
FloorRequestQuery =      (COMMON-HEADER)
                         (FLOOR-REQUEST-ID)
                        *[EXTENSION-ATTRIBUTE]
```

A.5.4 FloorRequestStatus

```
FloorRequestStatus =      (COMMON-HEADER)
                          (FLOOR-REQUEST-INFORMATION)
                         *[EXTENSION-ATTRIBUTE]
```

A.5.5 UserQuery

```
UserQuery =      (COMMON-HEADER)
                 [BENEFICIARY-ID]
                *[EXTENSION-ATTRIBUTE]
```

[38] Note that the EXTENSION-ATTRIBUTErefers to future extensions to BFCP protocol attributes.

A.5.6 UserStatus

```
UserStatus =        (COMMON-HEADER)
                    [BENEFICIARY-INFORMATION]
                   *(FLOOR-REQUEST-INFORMATION)
                   *[EXTENSION-ATTRIBUTE]
```

A.5.7 FloorQuery

```
FloorQuery =        (COMMON-HEADER)
                   *(FLOOR-ID)
                   *[EXTENSION-ATTRIBUTE]
```

A.5.8 FloorStatus

```
FloorStatus =       (COMMON-HEADER)
                  *1(FLOOR-ID)
                   *[FLOOR-REQUEST-INFORMATION]
                   *[EXTENSION-ATTRIBUTE]
```

A.5.9 ChairAction

```
ChairAction =       (COMMON-HEADER)
                    (FLOOR-REQUEST-INFORMATION)
                   *[EXTENSION-ATTRIBUTE]
```

A.5.10 ChairActionAck

```
ChairActionAck =    (COMMON-HEADER)
                   *[EXTENSION-ATTRIBUTE]
```

A.5.11 Hello

```
Hello =       (COMMON-HEADER)
             *[EXTENSION-ATTRIBUTE]
```

A.5.12 HelloAck

```
HelloAck = (COMMON-HEADER)
           (SUPPORTED-PRIMITIVES)
```

(SUPPORTED-ATTRIBUTES)
*[EXTENSION-ATTRIBUTE]

A.5.13 Error

Error = (COMMON-HEADER)
(ERROR-CODE)
[ERROR-INFO]
*[EXTENSION-ATTRIBUTE]

A.6 H.248 Descriptors

H.248 Messages may include various descriptors, given in detail in H.248.1 and as a summary in the table below [Itu05].

Descriptor name	**Description**
Modem	Identifies modem type and properties when applicable. (descriptor deprecated in ITU-T Rec. H.248.1 version 2 (05/2002))
Mux	Describes multiplex type for multimedia terminations (e.g., H.221, H.223, H.225.0) and terminations forming the input mux.
Media	A list of media stream specifications (see [Itu05] 7.1.4).
TerminationState	Properties of a termination (which can be defined in packages) that are not stream specific.
Stream	A list of Remote/Local/LocalControl Descriptors for a single stream.
Local	Contains properties that specify the media flows that the MG receives from the remote entity.
Remote	Contains properties that specify the media flows that the MG sends to the remote entity.
LocalControl	Contains properties (which can be defined in packages) that are of interest between the MG and the MGC.
Events	Describes events to be detected by the MG and what to do when an event is detected.
EventBuffer	Describes events to be detected by the MG when event buffering is active.
Signals	Describes signals (see [Itu05] 7.1.11) applied to terminations.
Audit	In Audit commands, identifies which information is desired.
Packages	In AuditValue, returns a list of packages realized by the termination.
DigitMap	Defines patterns against which sequences of a specified set of events are to be matched so they can be reported as a group rather than singly.
ServiceChange	In ServiceChange, what, why ServiceChange occurred, etc.
ObservedEvents	In Notify or AuditValue, report of events observed.
Statistics	In Subtract and Audit, report of statistics kept on a termination or stream.
Topology	Specifies flow directions between terminations in a context.
ContextAttribute	Contains properties (which can be defined in packages) that affect the context as a whole.
Error	Contains an Error Code and optionally error text; it may occur in command replies and in Notify requests.

A.7 H.248 Message Example: *Add*

H.248 Messages can be textual or binary encoded. The H.248 v3 message in this example is textual, carrying an *Add* command for one termination having one inactive stream with standard [HJP06] remote and local descriptors with H.248 deviations[39].

```
MEGACO/3 [192.168.55.54]
    Transaction = 1 {
            Context = $ {
                    Priority = 12,
                    Emergency,
                    Add = ip/104/1/$ {
                            Media {
                                    Stream = 1 {
                                            LocalControl {
                                                    mode = Inactive,
                                                    ds/dscp = 46,
                                                    gm/saf = ON,
                                                    gm/spf = ON,
                                                    eri_seco/rsb = FORWARD,
                                                    tman/sdr = 16000,
                                                    tman/pol = ON
                                            },
                                            Local {
                                                    v=0
                                                    c=IN IP4 $
                                                    m=- $ RTP/AVP -
                                                    b=AS:128
                                            },
                                            Remote {
                                                    v=0
                                                    c=IN IP4 25.196.80.72
                                                    m=- 20000 RTP/AVP -
                                                    b=AS:128
                                            }
                                    }
                            },
                            Signals {
                                    eri_seco/nap {
                                                    Stream = 1,
                                                    Action = LATCH
                                    }
                            },
                            Events = 1234 {
                                    g/cause,
                                    eri_seco/ms
                            }
                    }
            }
    }
```

[39] Not all SDP mandatory lines are mandatory in H.248 [Itu05].